

# **Studijní opora pro předmět Automatizace**

Study instrument for the subject automation

Petr Mičola

---

Bakalářská práce  
2011

 Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky

---

Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky  
akademický rok: 2010/2011

## ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Petr MIČOLA**  
Osobní číslo: **A08069**  
Studijní program: **B 3902 Inženýrská informatika**  
Studijní obor: **Informační a řídicí technologie**

Téma práce: **Studijní opora pro předmět Automatizace**

Zásady pro vypracování:

1. Seznamte se s problematikou probíranou v rámci přednášek předmětu Automatizace a se stávajícími výukovými podklady.
2. Po dohodě s vedoucím Bakalářské práce navrhnete formu zpracování studijních podkladů tak, aby byly dobře dostupné studentům a aby byly použitelné při vlastní výuce – zhodnoťte výhody a nevýhody.
3. Na základě dohody s vedoucím práce doplňte další kapitoly vyučované problematiky a realizujte je v prostředí PowerPointu.
4. Zpracujte ilustrativní řešené příklady do zadaných kapitol.
5. Zpracujte elektronickou pomůcku pro přednášky předmětu.

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. Švarc, I.: **Automatizace ? Automatické řízení**. Akademické nakladatelství CERM, s.r.o. Brno, 2005, ISBN 80-214-2943-7
2. Vítečková M., Víteček A. : **Základy automatické regulace**. VŠB ? Technická univerzita Ostrava, 2006
3. Balátě, J.: **Automatické řízení**, ISBN 80-7300-020-2, BEN-technická literatura, 2003
4. Franklin G.F., Powell J.D., Emami-Naeini A.: **Feedback control of dynamic systems**. Pearson Education, Inc, 2006
5. Vašek, V.: **Teorie automatického řízení II**. Skriptum FT VUT v Brně, Zlín 1990.
6. Švejda, J.: **Multimediální podpora výuky předmětu Automatizace**, BP, FAI UTB ve Zlíně
7. <http://www.e-automatizace>
8. Vašek, V.: **Pomůcka pro přednášky z předmětu Mikropočítače**. Interní pomůcka FT UTB, Zlín, 2005
9. FARANA, R. 1999. **Sylaby a cvičení k tvorbě Active Server Pages**. Ostrava: VŠB-TUO, 1999. Dostupný na URL: <http://www.fs.vsb.cz/books/ASPTest/Welcome.htm>.

Vedoucí bakalářské práce:

**prof. Ing. Vladimír Vašek, CSc.**

Ústav automatizace a řídicí techniky

Datum zadání bakalářské práce:

**25. února 2011**

Termín odevzdání bakalářské práce:

**7. června 2011**

Ve Zlíně dne 25. února 2011

prof. Ing. Vladimír Vašek, CSc.  
*děkan*



prof. Ing. Vladimír Vašek, CSc.  
*ředitel ústavu*

## **ABSTRAKT**

V této bakalářské práci je nejprve popsána problematika základních logických funkcí a Laplaceovy transformace. Dále je zmíněno téma linearity a popis základních dynamických lineárních členů. Pro syntézu regulačního obvodu je představena metoda požadovaného modelu. Zvláštní zájem je udělen programovacímu jazyku Java a jeho využití pro aplikaci, která má za úkol animačně vykreslovat průběhy přechodových charakteristik zadaných přenosů. Hlavním cílem je vylíčit zmíněná témata a popsat postup při řešení příkladů.

Klíčová slova: automatizace, Laplaceova transformace, dynamický lineární člen, syntéza, regulace, regulátor, jazyk Java, animace, metoda požadovaného modelu, řešený příklad

## **ABSTRACT**

Firstly the issues of basic logic functions and Laplace transform are described in this Bachelor thesis. Furthermore topic of linearity and description of basic dynamic linear members are mentioned. Method of required model is introduced. The model is example for synthesis of closed feedback loop. Special interest is given to programming language Java and its usage for application, which animates step response of specified transmissions. The main goal is to described mentioned topics and solved examples.

Keywords: automation, Laplace transform, dynamic linear member, synthesis, regulation, regulator, Java language, animation, method of required model, solved example

### **Poděkování**

Tímto bych rád poděkoval prof. Ing. Vladimíru Vaškovi CSc., který mi jeho odbornými radami a věcnými připomínkami přispěl vypracovat tuto bakalářskou práci. Děkuji také Ing. Tomáši Dulíkovi za velmi cenné programátorské návrhy.

### **Motto**

„Štěstí přeje silným a statečným.“

**Prohlašuji, že**

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

**Prohlašuji,**

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....

podpis diplomanta

**OBSAH**

<b>ÚVOD</b> .....	<b>9</b>
<b>I TEORETICKÁ ČÁST</b> .....	<b>10</b>
<b>1 LOGICKÉ FUNKCE</b> .....	<b>11</b>
1.1 ZÁKLADNÍ LOGICKÉ FUNKCE .....	11
1.2 ZPŮSOBY VYJÁDŘENÍ .....	13
1.3 MINIMALIZACE LOGICKÝCH FUNKCÍ .....	14
<b>2 LAPLACEOVA TRANSFORMACE</b> .....	<b>18</b>
2.1 ZÁKLADNÍ DEFINICE .....	18
2.2 ZÁKLADNÍ VĚTY LAPLACEOVY TRANSFORMACE .....	19
2.3 ROZKLAD NA PARCIÁLNÍ ZLOMKY .....	19
2.4 HEAVISIDEŮV ROZVOJ .....	20
<b>3 LINEARITA</b> .....	<b>22</b>
3.1 LINEARIZACE TEČNOU .....	22
<b>4 ZÁKLADNÍ DYNAMICKÉ LINEÁRNÍ ČLENY</b> .....	<b>24</b>
4.1 PROPORCIONÁLNÍ ČLEN .....	24
4.2 INTEGRAČNÍ ČLEN .....	26
4.3 DERIVAČNÍ ČLEN .....	27
<b>5 SYNTÉZA REGULAČNÍHO OBVODU</b> .....	<b>29</b>
5.1 STRUKTURA REGULÁTORU .....	29
5.2 KVALITA REGULACE.....	30
5.3 INTEGRÁLNÍ KRITÉRIA .....	34
5.3.1 Lineární regulační plocha.....	34
5.3.2 Absolutní regulační plocha.....	35
5.3.3 Kvadratická regulační plocha.....	36
5.3.4 Kritérium ITAE .....	37
5.4 METODA POŽADOVANÉHO MODELU.....	38
<b>6 PROGRAMOVACÍ JAZYK JAVA</b> .....	<b>42</b>
6.1 ZÁKLADY PROGRAMOVACÍHO JAZYKU JAVA .....	42
6.1.1 Datové typy proměnných.....	43
6.1.2 Řídící příkazy .....	43
6.1.3 Výjimky.....	46
6.1.4 Vlákna .....	46
6.1.5 GUI.....	48
6.2 VÝVOJOVÉ PROSTŘEDÍ ECLIPSE 3.5 .....	48
<b>II PRAKTICKÁ ČÁST</b> .....	<b>50</b>
<b>7 LOGICKÉ FUNKCE</b> .....	<b>51</b>

7.1	KANONICKÉ TVARY – ŘEŠENÝ PŘÍKLAD.....	51
7.2	KARNAUGHOVA MAPA – ŘEŠENÝ PŘÍKLAD .....	52
7.3	QUINEOVA-McCLUSKEYOVA METODA – ŘEŠENÝ PŘÍKLAD .....	52
<b>8</b>	<b>LAPLACEOVA TRANSFORMACE.....</b>	<b>54</b>
8.1	LAPLACEOVA TRANSFORMACE – ŘEŠENÝ PŘÍKLAD 1 .....	54
8.2	LAPLACEOVA TRANSFORMACE – ŘEŠENÝ PŘÍKLAD 2 .....	54
8.3	LAPLACEOVA TRANSFORMACE – ŘEŠENÝ PŘÍKLAD 3 .....	55
8.4	LAPLACEOVA TRANSFORMACE – ŘEŠENÝ PŘÍKLAD 4 .....	56
8.5	LAPLACEOVA TRANSFORMACE – ŘEŠENÝ PŘÍKLAD 5 .....	56
8.6	LAPLACEOVA TRANSFORMACE – ŘEŠENÝ PŘÍKLAD 6 .....	57
8.7	LAPLACEOVA TRANSFORMACE – ŘEŠENÝ PŘÍKLAD 7 .....	60
8.8	LAPLACEOVA TRANSFORMACE – ŘEŠENÝ PŘÍKLAD 8 .....	60
8.9	LAPLACEOVA TRANSFORMACE – ŘEŠENÝ PŘÍKLAD 9 .....	61
<b>9</b>	<b>METODA POŽADOVANÉHO MODELU .....</b>	<b>62</b>
9.1	METODA POŽADOVANÉHO MODELU – ŘEŠENÝ PŘÍKLAD .....	62
<b>10</b>	<b>APLIKACE NA VYKRESLOVÁNÍ GRAFŮ .....</b>	<b>64</b>
10.1	PROPORCIONÁLNÍ ČLEN .....	64
10.2	INTEGRAČNÍ ČLEN .....	76
	<b>ZÁVĚR .....</b>	<b>77</b>
	<b>ZÁVĚR V ANGLIČTINĚ .....</b>	<b>78</b>
	<b>SEZNAM POUŽITÉ LITERATURY.....</b>	<b>79</b>
	<b>SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK .....</b>	<b>81</b>
	<b>SEZNAM OBRÁZKŮ .....</b>	<b>82</b>
	<b>SEZNAM TABULEK.....</b>	<b>83</b>
	<b>SEZNAM PŘÍLOH.....</b>	<b>84</b>

## ÚVOD

Velmi často využívanou pomůckou při sdělování informací většímu množství posluchačů jsou elektronické prezentace. Díky této pomůcce má přednášející i posluchači k dispozici oporu, která stručnými body vystihuje danou problematiku. Výhodné je poskytnout před začátkem prezentace posluchačům přednášené materiály. Vzhledem ke stručné osnově, kterou prezentace obsahují, si mohou posluchači připsávat vlastní poznámky přímo k jednotlivým bodům a tím lépe pochopit rozebíraná témata. Elektronické prezentace disponují řadou možností, jak zefektivnit přednášený výklad. Základem mohou být jednoduché úpravy jako tučné písmo či podtržení. Složitější efekty jako animace a vkládání cizích objektů do prezentací zdůrazňují řešený problém.

Cizím objektem vloženým do prezentace může být jakýkoli soubor. V této bakalářské práci jsem využil univerzálnosti programovacího jazyku Java. Do prezentace byly vloženy aplikace vytvořené v tomto programovacím jazyku. Tyto aplikace jsou běžnými spustitelnými soubory. Po jejich spuštění je otevřeno nové okno s běžící aplikací. Jakmile je aplikace ukončena, tak je uživatel vrácen zpět na původní snímek. Vkládáním a spouštěním cizích souborů do prezentace lze docílit efektů, které standardní program pro vytváření elektronických prezentací neobsahuje.

Tato bakalářská práce doplňuje a rozšiřuje již existující prezentace, které slouží jako opora pro předmět Automatizace. Do této opory byly přidány nové kapitoly a ilustrativní řešené příklady ke zvoleným tématům. Byla vytvořena aplikace simulující průběh přechodových charakteristik proporcionalních a integračních členů.

## **I. TEORETICKÁ ČÁST**

## 1 LOGICKÉ FUNKCE

Při logickém řízení se v logickém obvodu zpracovávají informace o procesu, který je řízen. Pomocí těchto informací jsou ovládána zařízení za účelem splnění zadaného úkolu.

Logickým obvodem rozumíme fyzikální systém, jež se skládá z logických prvků vzájemně propojených mezi sebou logickými veličinami.

K popisu spojitých veličin využíváme spojité proměnné, které mohou dosahovat nekonečného počtu hodnot. Logické proměnné mohou dosahovat pouze konečného počtu hodnot. Logická algebra je soustava pravidel, které popisují vztahy mezi logickými proměnnými.

Dvuhodnotové proměnné (veličiny) jsou logické proměnné, jež mohou dosáhnout pouze dvou hodnot (nejrozšířenější je používání hodnot 0 a 1). Tyto hodnoty mohou v reálném životě vyjadřovat například dvojice pravda – nepravda, proud teče – neteče, zařízení běží – neběží, teplota je nad stanovenou mezí – teplota není nad stanovenou mezí, atd. [6]

Logickou funkcí rozumíme funkci danou vztahem:

$$y = f(x_1, x_2, \dots, x_n) \quad (1)$$

Logická proměnná  $y$  je závislá na kombinaci proměnných  $x_1, \dots, x_n$ , které jsou nezávislé.

### 1.1 Základní logické funkce

Mezi základní logické funkce patří logické funkce jedné proměnné:

$$y = f(x) \quad (2)$$

Logická funkce jedné proměnné má čtyři pravdivostní tabulky, které určují závislost proměnné  $y$  na proměnné  $x$ .

$x$	$y$
0	0
1	0

falsum

$x$	$y$
0	1
1	1

verum

$x$	$y$
0	0
1	1

aserce

$x$	$y$
0	1
1	0

negace

Tabulka 1: Pravdivostní tabulky logické funkce jedné proměnné

V pravdivostních tabulkách je vidět, že funkce falsum má vždy hodnotu výstupu  $y$  rovnu 0, bez ohledu na to, jakou má hodnotu vstupu  $x$ . Funkce verum nabývá hodnoty výstupní proměnné  $y$  vždy 1 při jakémkoli vstupu  $x$ . Pokud je na výstup  $y$  kopírován vstup  $x$ , pak se jedná o aserci. Nejvýznamnější z těchto čtyř funkcí je funkce označená jako negace. Tato funkce má výstupní proměnnou  $y$  rovnu opačné hodnotě vstupní proměnné  $x$ . Negace se označuje:

$$y = \bar{x} \quad (3)$$

Logická funkce dvou proměnných může vypadat například takto:

$$y = f(x_1, x_2) \quad (4)$$

Mezi nejvýznamnější logické funkce dvou proměnných patří logický součet (OR), logický součin (AND), negace logického součtu (NOR) a negace logického součinu (NAND).

V následující tabulce jsou zobrazeny pravdivostní tabulky jednotlivých funkcí:

OR			AND			NOR			NAND		
$x_1$	$x_2$	$y$	$x_1$	$x_2$	$y$	$x_1$	$x_2$	$y$	$x_1$	$x_2$	$y$
0	0	0	0	0	0	0	0	1	0	0	1
0	1	1	0	1	0	0	1	0	0	1	1
1	0	1	1	0	0	1	0	0	1	0	1
1	1	1	1	1	1	1	1	0	1	1	0

Tabulka 2: Pravdivostní tabulky logických funkcí OR, AND, NOR, NAND

Logický součet (OR) se také nazývá disjunkce a platí pro něj pravidlo, že pokud je alespoň jeden ze vstupů  $x_1, x_2$  roven 1, pak je výstupní veličina  $y$  rovna 1.

Logický součin (AND) se označuje jako konjunkce. Výstupní proměnná  $y$  je rovna 1 pouze v případě, kdy jsou obě vstupní proměnné  $x_1, x_2$  rovny 1.

Negace logického součtu (NOR) se někdy nazývá Pierceova funkce. Výstupní proměnná  $y$  je rovna 1, pouze pokud jsou vstupní proměnné  $x_1, x_2$  rovny 0.

Negace logického součinu (NAND) se někdy označuje jako Shefferova funkce. Platí pro ni, že výstupní proměnná  $y$  je rovna 1 v případě, že vstupní proměnné  $x_1, x_2$  nejsou zároveň 1.

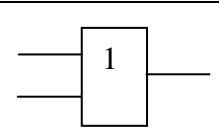
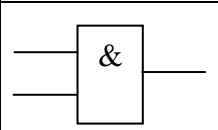
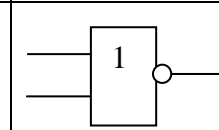
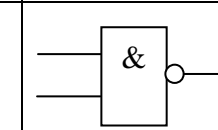
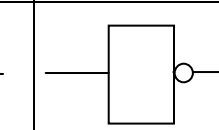
## 1.2 Způsoby vyjádření

Logické funkce lze vyjádřit několika způsoby. Jedním z nejzákladnějších způsobů vyjádření je slovní zadání funkce.

Dalším možným způsobem je zapsání logické funkce pomocí pravdivostní tabulky. V tabulce jsou zapsány všechny kombinace vstupních proměnných  $x_1, \dots, x_n$  a těmto kombinacím je přiřazena hodnota výstupní proměnné  $y$ . Podle toho kolik má funkce vstupů  $x_1, \dots, x_n$ , tak tím je určen počet řádků tabulky, který je  $2^n$ . Příklad pravdivostní tabulky funkce OR je například v tabulce Tabulka 2: Pravdivostní tabulky logických funkcí OR, AND, NOR, NAND.

Z pravdivostní tabulky se dá určit další způsob zápisu logické funkce. Tímto způsobem je zápis pomocí algebraického výrazu, kdy jednotlivé řádky tabulky, ve kterých je výstupní proměnná  $y$  rovna 1, jsou zapsány pomocí součinů vstupních proměnných  $x_1, \dots, x_n$ . V případě, že je vstupní proměnná rovna logické 1, tak se pouze opíše její název. Pokud má vstupní proměnná hodnotu logická 0, pak se při zápisu do součinu musí tato proměnná znegovat. Jednotlivé součiny vyjadřující řádky se pak sečtou a výsledkem je součet součinů vstupních proměnných  $x_1, \dots, x_n$ . [6]

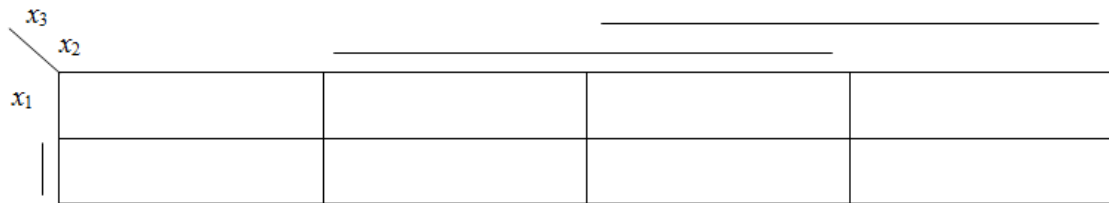
Blokové schéma je další možností, jak vyjádřit logickou funkci. Jak již název napovídá, jedná se o kombinaci bloků, do kterých vstupují vstupní proměnné a vystupují z nich výstupní proměnné. Kombinací základních bloků (OR, AND, NOR, NAND, NON) lze vyjádřit i složitější logické funkce. Základní schematické značky mají tvar uvedený v následující tabulce:

OR	AND	NOR	NAND	NON
				

Tabulka 3: Blokové schéma logických funkcí OR, AND, NOR, NAND, NON

Složitější logické funkce, které jsou vyjádřeny pomocí blokových schémat, mohou být nepřehledné. To je jedním z důvodů, proč je lepší logickou funkci minimalizovat. K minimalizaci a popisu slouží Karnaughova mapa. Tato mapa je vlastně tabulka, která má tolik buněk, kolik je možných kombinací vstupních proměnných. Pokud tedy má logická funkce tři proměnné  $x_1, x_2, x_3$ , pak je počet buněk tabulky  $2^3 = 8$ . Do těchto buněk se pak

zapisuje hodnota výstupní proměnné  $y$ . U Karnaughovy mapy se sousední buňky vstupních proměnných odlišují v jedné hodnotě. Vedle řádku a sloupce napíšeme název vstupní proměnné. Svislou nebo vodorovnou čarou je vyjádřena hodnota vstupní proměnné. Příklad Karnaughovy mapy pro tři vstupní proměnné  $x_1, x_2, x_3$ :



Obrázek 1: Karnaughova mapa pro tři vstupní proměnné

### 1.3 Minimalizace logických funkcí

Logické funkce mohou být vyjádřeny několika způsoby, ale vždy jsou matematicky rovnocenné. Což ovšem neznamená, že jsou uvedeny v nejjednodušším tvaru. Pokud lze logickou funkci zjednodušit na tvar, který již nelze zjednodušit, dosáhli jsme minimalizovaného tvaru logické funkce. Tento tvar je výhodnější používat zejména proto, že je při něm použito méně logických prvků (OR, AND, NON), než kdyby byl uvažován tvar logické funkce, který lze ještě minimalizovat.

Jednou z možných metod je algebraická minimalizace. Tato metoda využívá pravidla Booleovy algebry. Avšak pokud máme složitější funkce, tak je tato metoda časově náročná.

[6]

Základní pravidla Booleovy algebry:

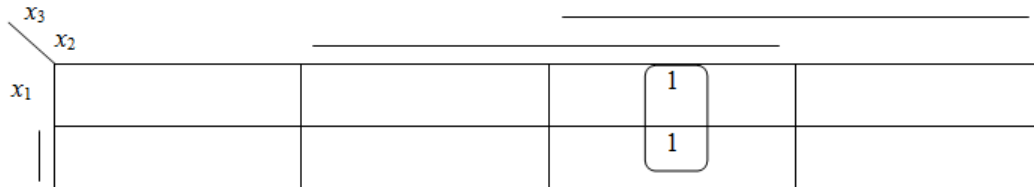
$1 * 1 = 1$
$1 * 0 = 0 * 1 = 0$
$0 * 0 = 0$
$0 + 0 = 0$
$0 + 1 = 1 + 0 = 1$
$1 + 1 = 1$
$\overline{0} = 1$
$\overline{1} = 0$
$\overline{\overline{0}} = 0$

Obrázek 2: Základní pravidla  
Booleovy algebry

<b>Zákon vyloučení třetího</b>	$x + \bar{x} = 1$
<b>Logický rozpor</b>	$x * \bar{x} = 1$
<b>Zákon opakování</b>	$x + x = x$ $x * x = x$
<b>Komutativní zákony</b>	$x_1 + x_2 = x_2 + x_1$ $x_1 * x_2 = x_2 * x_1$
<b>Asociativní zákony</b>	$(x_1 + x_2) + x_3 = x_1 + x_2 + x_3$ $(x_1 * x_2) * x_3 = x_1 * x_2 * x_3$
<b>Distributivní zákony</b>	$x_1 + x_2 * x_3 = (x_1 + x_2) * (x_1 + x_3)$ $x_1 *(x_2 + x_3) = x_1 * x_2 + x_1 * x_3$
<b>Absorbční zákony</b>	$x_1 + x_1 * x_2 = x_1$ $x_1 *(x_1 + x_2) = x_1$ $x_1 + \bar{x}_1 * x_2 = x_1 + x_2$ $x_1 *( \bar{x}_1 + x_2 ) = x_1 * x_2$
<b>Neutrálnost</b>	$0 + x = x$ $1 + x = 1$ $1 * x = x$ $0 * x = 0$
<b>De Morganovy zákony</b>	$\overline{x_1 + x_2} = \bar{x}_1 * \bar{x}_2$ $\overline{x_1 * x_2} = \bar{x}_1 + \bar{x}_2$

Tabulka 4: Základní pravidla Booleovy algebry

Jak bylo zmíněno v minulé kapitole, tak k minimalizaci logických funkcí se dají využívat Karnaughovy mapy. Logická funkce je minimalizována tak, že jsou do tabulky (Karnaughovy mapy) zapsány hodnoty jednotlivých kombinací. Sousední buňky obsahující logickou 1 se dají sloučit do dvojic, čtveřic, osmic, atd.



Obrázek 3: Karnaughova mapa funkce  $y=x_2x_3$

Logická funkce má pak tvar dle příslušných dvojic, čtveřic, osmic, atd. Přičemž platí pravidlo, že v této dvojici, čtveřici, osmici, atd. chybí ta vstupní proměnná, která se mění. Např. minimalizovaná funkce z obrázku Obrázek 3: Karnaughova mapa funkce  $y=x_2x_3$  má tvar:

$$y = x_2x_3 \quad (5)$$

U této funkce se mění řádek pro vstupní proměnnou  $x_1$ , tudíž výstupní funkce ji neobsahuje. Při slučování jedniček v tabulce musí být každá jednička zakroužkována. Platí také, že každá jednička může být označena vícekrát. To znamená, že jedna jednička může být součástí dvojice, ale také třeba jiné čtveřice. U Karnaughovy mapy je cílem, provést co nejméně smyček. Čím méně smyček, tím je minimalizace účinnější. [6]

## 2 LAPLACEOVA TRANSFORMACE

Laplaceova transformace vznikla již v roce 1820 za účelem jednoduššího řešení diferenciálních rovnic. Vzhledem k tomu, že řešení diferenciální rovnice je poměrně obtížné, tak se využívá převod pomocí Laplaceovy transformace do prostoru obrazů. V tomto prostoru obrazů je již řešení diferenciální rovnice mnohem jednodušší. Rovnice se vyřeší a pak se zpět převede pomocí zpětné Laplaceovy transformace do původního prostoru originálů. Prostor obrazů je oblast komplexní proměnné a prostor originálů je oblast časové proměnné. Laplaceova transformace se také využívá pro popis lineárních spojitých regulačních systémů. [4], [6], [10], [14]

### 2.1 Základní definice

Transformace je přiřazení obrazu  $F(s)$  originální funkci  $f(t)$ . Přímá transformace je tedy zobrazení funkce reálné proměnné  $f(t)$ , která je v prostoru originálů, na funkci komplexní proměnné  $F(s)$ , která se nachází v prostoru obrazů. Zpětná transformace je převod funkce komplexní proměnné  $F(s)$  z prostoru obrazů na funkci reálné proměnné  $f(t)$  do prostoru originálů.

Přímá Laplaceova transformace je dána vztahem:

$$F(s) = L\{f(t)\} = \int_0^{\infty} f(t)e^{-st} dt \quad (6)$$

Zpětná Laplaceova transformace je dána vztahem:

$$f(t) = L^{-1}\{F(s)\} = \frac{1}{2\pi j} \int_{c-j\infty}^{c+j\infty} F(s)e^{st} ds \quad (7)$$

V uvedených vztazích je  $t$  reálná proměnná,  $s$  je komplexní proměnná,  $c$  je považováno za konstantu,  $j$  je imaginární jednotka.

[7], [10]

## 2.2 Základní věty Laplaceovy transformace

V následujících vztazích je uvažováno  $t$  jako reálná proměnná,  $s$  jako komplexní proměnná,  $c_1$  a  $c_2$  jsou za konstanty,  $n$  značí stupeň derivace,  $a$  je konstanta.

Věta o linearitě:

$$L\{c_1 f(t) + c_2 g(t)\} = L\{c_1 f(t)\} + L\{c_2 g(t)\} = c_1 F(s) + c_2 G(s) \quad (8)$$

Věta o obrazu  $n$ -té derivace:

$$L\{f^{(n)}(t)\} = s^n F(s) - s^{n-1} f(0) - s^{n-2} f'(0) - \dots - s^0 f^{(n-1)}(0) \quad (9)$$

Věta o obrazu integrálu:

$$L\left\{\int_0^t f(t) dt\right\} = \frac{1}{s} F(s) \quad (10)$$

Věta o počáteční hodnotě:

$$\lim_{t \rightarrow 0} f(t) = \lim_{s \rightarrow \infty} sF(s) \quad (11)$$

Věta o koncové hodnotě:

$$\lim_{t \rightarrow \infty} f(t) = \lim_{s \rightarrow 0} sF(s) \quad (12)$$

Věta o posunutí:

$$L\{f(t-a)\} = e^{-as} F(s) \quad (13)$$

[6], [2], [14]

## 2.3 Rozklad na parciální zlomky

Slovník Laplaceovy transformace obsahuje nejzákladnější a nejběžnější převody mezi originály a obrazy. Tímto značně usnadňuje převod mezi oblastí originálů a oblastí obrazů. Nejčastěji vzniká problém při zpětné transformaci, kdy mají obrazy tvar, který není ve slovníku. V takovém případě je nutné obraz upravit na vhodný tvar, který se již ve slovníku objevuje. V praxi se velmi často používá metoda rozkladu na parciální zlomky.

Nejčastější tvar obrazu ryzí racionální lomené funkce:

$$F(s) = \frac{M(s)}{N(s)} = \frac{b_m s^m + \dots + b_1 s^1 + b_0 s^0}{a_n s^n + \dots + a_1 s^1 + a_0 s^0}, \quad n > m \quad (14)$$

Pokud není splněna podmínka  $n > m$ , pak je nutné vydělit polynom  $M(s)$  polynomem  $N(s)$ .

Polynom ve jmenovateli  $N(s)$  lze rozložit na tvar:

$$N(s) = a_n s^n + \dots + a_1 s + a_0 = a_n (s - s_1)(s - s_2) \dots (s - s_n) \quad (15)$$

Kořeny jmenovatele  $s_1, s_2, \dots, s_n$  nazýváme také póly. Kořeny jmenovatele mohou být reálné nebo komplexní, jednonásobné nebo mnohonásobné.

Je-li kořen  $s_1$  reálný jednonásobný a kořen  $s_3$  reálný  $r$ -násobný, pak bude rozklad vypadat následovně:

$$F(s) = \frac{M(s)}{N(s)} = \frac{M(s)}{(s - s_1)(s - s_2)^r} = \frac{A}{s - s_1} + \frac{B_1}{s - s_2} + \frac{B_2}{(s - s_2)^2} + \dots + \frac{B_r}{(s - s_2)^r} \quad (16)$$

Je-li dvojice komplexních kořenů  $s^2 + as + b$  jednonásobná a dvojice komplexních kořenů  $s^2 + cs + d$   $q$ -násobná, pak bude rozklad vypadat následovně:

$$F(s) = \frac{M(s)}{N(s)} = \frac{M(s)}{(s^2 + as + b)(s^2 + cs + d)^r} \quad (17)$$

$$F(s) = \frac{As + B}{s^2 + as + b} + \frac{C_1 s + D_1}{s^2 + cs + d} + \frac{C_2 s + D_2}{(s^2 + cs + d)^2} + \dots + \frac{C_q s + D_q}{(s^2 + cs + d)^q}$$

Příslušné konstanty  $A, B, B_1, \dots, B_r, C_1, \dots, C_r, D_1, \dots, D_r$  se určují dosazovací metodou nebo metodou neurčitých koeficientů. Při složitějších výpočtech je dobré tyto metody kombinovat. [2], [10]

## 2.4 Heavisideův rozvoj

Alternativou k metodě rozkladu na parciální zlomku je rozklad pomocí Heavisideova rozvoje. Opět se vychází ze tvaru pro ryzí racionální funkci:

$$F(s) = \frac{M(s)}{N(s)} = \frac{b_m s^m + \dots + b_1 s + b_0}{a_n s^n + \dots + a_1 s + a_0}, \quad n > m \quad (18)$$

Polynom ve jmenovateli  $N(s)$  lze rozložit na součin kořenových činitelů. Pak lze zadanou funkci rozložit na požadovaný tvar.

Pokud jsou kořeny jednonásobné, lze funkci rozložit na následující tvar:

$$F(s) = \frac{A_1}{s-s_1} + \frac{A_2}{s-s_2} + \dots + \frac{A_n}{s-s_n} \quad (19)$$

Čitatele jednotlivých zlomků  $A_i$  se pak určují podle vztahu:

$$A_i = \lim_{s \rightarrow s_i} [(s-s_i)F(s)] = [(s-s_i)F(s)]_{s=s_i} \quad (20)$$

Pokud jsou kořeny  $r$ -násobné, lze funkci rozložit na tvar:

$$F(s) = \frac{B_1}{s-s_1} + \frac{B_2}{(s-s_1)^2} + \dots + \frac{B_r}{(s-s_1)^r} \quad (21)$$

Čitatele jednotlivých zlomků  $B_i$  se pak určují podle vztahu:

$$\begin{aligned} B_r &= [(s-s_1)^r F(s)]_{s=s_1} \\ B_{r-1} &= \frac{1}{1!} \left[ \frac{d}{ds} \{(s-s_1)^r F(s)\} \right]_{s=s_1} \\ B_{r-2} &= \frac{1}{2!} \left[ \frac{d^{(2)}}{ds^{(2)}} \{(s-s_1)^r F(s)\} \right]_{s=s_1} \\ &\vdots \\ B_1 &= \frac{1}{(r-1)!} \left[ \frac{d^{(r-1)}}{ds^{(r-1)}} \{(s-s_1)^r F(s)\} \right]_{s=s_1} \end{aligned} \quad (22)$$

[1], [7], [4]

### 3 LINEARITA

Systém, který se skládá pouze z lineárních členů, se označuje jako systém lineární. Statickou charakteristikou takového systému je přímka. Avšak pokud systém obsahuje jeden nebo více nelineárních členů, pak se jedná o systém nelineární, který má nelineární statickou charakteristiku. Vzájemná záměna členů v lineárním obvodu nemá vliv na odezvu. Tato skutečnost u nelineárních systémů neplatí. S lineárními systémy se velmi dobře pracuje a teorie lineární regulace je dobře promyšlená. Bohužel v praxi je většina systémů nelineárních. Díky dobré práci s lineárními systémy je snahou nelineární systémy linearizovat. [1], [7], [8], [10]

#### 3.1 Linearizace tečnou

Linearizací se rozumí popsat nelineární systém v okolí pracovního bodu lineárním matematickým modelem.

Systém s jedním výstupem a vstupy  $u_1, \dots, u_m$  je popsán následujícím vztahem:

$$y = f(u_1, \dots, u_m) \quad (23)$$

Pracovní bod je dán vztahem:

$$y_0 = f(u_{10}, \dots, u_{m0}) \quad (24)$$

V pracovním bodu se provede náhrada pomocí tečné nadroviny a získá se výstupní veličina v absolutním vyjádření po linearizaci.

$$\hat{y} = y_0 + \Delta y \quad (25)$$

Kde platí rovnice pro přírůstek výstupní veličiny.

$$\Delta y = k_1 \Delta u_1 + \dots + k_m \Delta u_m = y - y_0 \quad (26)$$

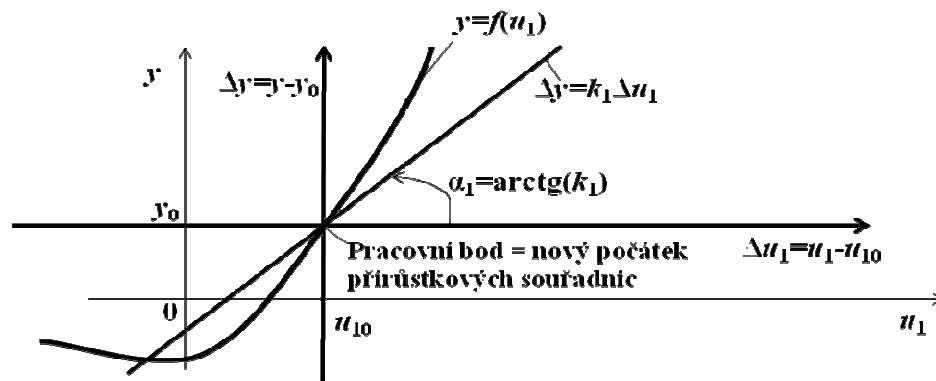
A také platí přírůstek pro vstupní veličiny.

$$\Delta u_1 = u_1 - u_{10}, \dots, \Delta u_m = u_m - u_{m0} \quad (27)$$

Parciální derivace v pracovním bodě jsou dány vztahy:

$$k_1 = \left. \frac{\partial f}{\partial u_1} \right|_0, \dots, k_m = \left. \frac{\partial f}{\partial u_m} \right|_0 \quad (28)$$

Pokud existují parciální derivace v pracovním bodu a jsou spojité, pak lze použít metodu linearizace tečnou hadrovinou. Po provedení linearizace se nový počátek souřadnicového systému přesune do pracovního bodu  $y_0$ . Linearizovaný model lze pak uvažovat jen pokud jsou změny vstupního signálu v okolí pracovního bodu malé. Jinak by byla linearizace nekvalitní. Složitější matematické vztahy je vhodné před linearizací rozdělit a linearizovat jednotlivé členy. [1], [10]



Obrázek 4: Geometrická interpretace linearizace tečnou

## 4 ZÁKLADNÍ DYNAMICKÉ LINEÁRNÍ ČLENY

Základní třídění těchto členů se posuzuje podle ustáleného stavu, který existuje nebo neexistuje. Pokud existuje, tak může být buď nulový nebo nenulový. Z průběhu přechodové charakteristiky se velmi snadno určí typ ustáleného stavu, neboť je vidět, na jaké hodnotě se ustálí. Hodnotu, kde se přechodová charakteristika ustálí, lze také určit z věty o konečné hodnotě:

$$h(\infty) = \lim_{t \rightarrow \infty} h(t) = \lim_{s \rightarrow 0} sH(s) = \lim_{s \rightarrow 0} s \frac{G(s)}{s} = \lim_{s \rightarrow 0} G(s) \quad (29)$$

[1], [6], [10]

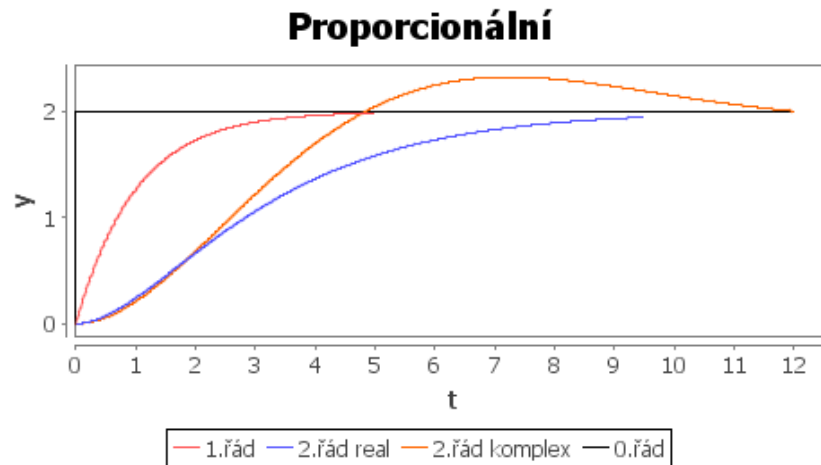
### 4.1 Proporcionální člen

Ustálený stav přechodové charakteristiky proporcionálních členů je nenulový. U těchto členů nelze vytknout komplexní proměnnou  $s$  ani z čitatele ani z jmenovatele. Proporcionální členy mohou být se setrvačností nultého až  $n$ -tého řádu.

Obecný proporcionální člen se setrvačností  $n$ -tého řádu a s dopravním zpožděním je dán rovnicí:

$$G(s) = \frac{b_m s^m + \dots + b_1 s + b_0}{a_n s^n + \dots + a_1 s + a_0} e^{-sT_d}, \quad n \geq m \quad (30)$$

Přechodové charakteristiky pak můžou vypadat následovně:



Obrázek 5: Proporcionální členy – přechodové charakteristiky

Proporcionální člen bez setrvačnosti je také označován jako ideální proporcionální člen nebo také proporcionální člen se setrvačností 0. řádu. Přenos takového členu je dán vztahem:

$$G(s) = k_0 \quad (31)$$

Pokud je koeficient  $|k_0| > 1$  pak se jedná o zesílení a pro  $|k_0| < 1$  se jedná o tlumení.

Proporcionální člen se setrvačností 1. řádu (aperiodický člen 1. řádu, setrvačný člen 1. řádu, reálný proporcionální člen) je dán vztahem:

$$G(s) = \frac{k_0}{T_1 s + 1} \quad (32)$$

$T_1$  je označována jako časová konstanta.

Proporcionální člen se setrvačností 2. řádu je dán vztahem:

$$G(s) = \frac{k_0}{T_0^2 s^2 + 2\zeta T_0 s + 1} \quad (33)$$

Podle hodnoty relativního tlumení  $\zeta$  je proporcionální člen se setrvačností 2. řádu aperiodický, mezní aperiodický, kmitavý nebo konzervativní.

Pro  $\zeta > 1$  se jedná o aperiodický člen:

$$G(s) = \frac{k_0}{(T_1s + 1)(T_2s + 1)} \quad (34)$$

Pro  $\zeta = 1$  se jedná o mezní aperiodický člen:

$$G(s) = \frac{k_0}{(T_0s + 1)^2} \quad (35)$$

Pro  $0 < \zeta < 1$  má jmenovatel přenosu  $G(s)$  komplexní kořeny a jedná se o kmitavý člen.

Pro  $\zeta = 0$  se jedná o konzervativní (bezztrátový) člen, který je na mezi stability a platí pro něj vztah:

$$G(s) = \frac{k_0}{T_0^2 s^2 + 1} \quad (36)$$

[1], [6], [10]

## 4.2 Integrační člen

V případě integračního členu neexistuje ustálený stav přechodové charakteristiky. Tyto členy bývají také označovány jako astatické členy. Charakteristickým rysem je možnost vytknout ze jmenovatele přenosu komplexní proměnnou  $s$ .

Integrační člen bez setrvačnosti je dán vztahem:

$$G(s) = \frac{k_0}{s} \quad (37)$$

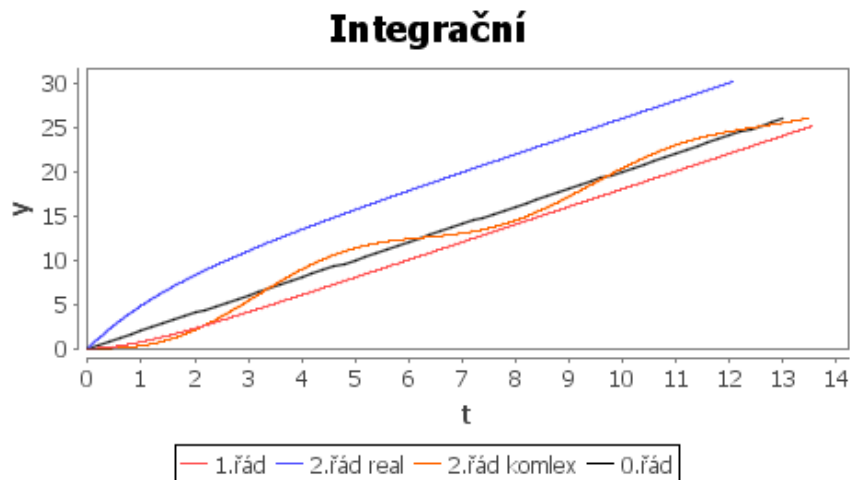
Integrační člen se setrvačností 1. řádu, také označovaný jako reálný integrační člen lze popsat vztahem:

$$G(s) = \frac{k_0}{s(T_1s + 1)} \quad (38)$$

Obecný integrační člen  $q$ -tého řádu se setrvačností  $n$ -tého řádu s dopravním zpožděním je popsán vztahem:

$$G(s) = \frac{b_m s^m + \dots + b_1 s + b_0}{s^q (a_n s^n + \dots + a_1 s + a_0)} e^{-sT_d}, \quad n + q \geq m \quad (39)$$

Průběhy přechodových charakteristik integračních členů mohou vypadat například následovně:



Obrázek 6: Integrační členy – přechodové charakteristiky

### 4.3 Derivační člen

U derivačního členu existuje ustálený stav přechodové charakteristiky, avšak je roven nule. Derivační členy se také označují jako předstihové členy. Charakteristickým rysem je možnost vytknout z čitatele přenosu komplexní proměnnou  $s$ .

Derivační člen bez setrvačnosti je označován také jako ideální derivační člen. Tento člen je ovšem fyzikální nerealizovatelný a má tvar:

$$G(s) = k_0 s \quad (40)$$

Derivační člen se setrvačností 1. řádu, označovaný jako reálný derivační člen, je dán vztahem

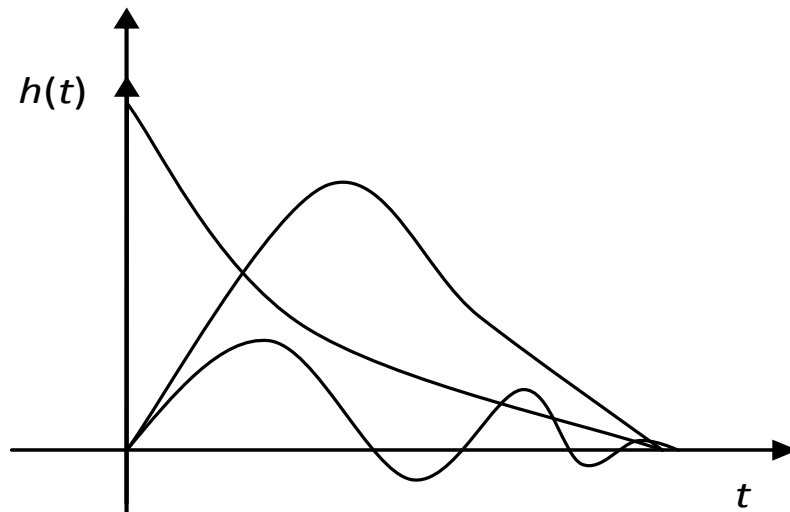
$$G(s) = \frac{k_0 s}{T_1 s + 1} \quad (41)$$

Obecný derivační člen  $q$ -tého řádu se setrvačností  $n$ -tého řádu s dopravním zpožděním lze zapsat vztahem:

$$G(s) = s^q \frac{b_m s^m + \dots + b_1 s + b_0}{a_n s^n + \dots + a_1 s + a_0} e^{-sT_d}, \quad n \geq m + q \quad (42)$$

[1], [6], [10]

Průběh přechodových charakteristik derivačních členů by mohl vypadat následovně:



Obrázek 7: Derivační členy – přechodové charakteristiky

## 5 SYNTÉZA REGULAČNÍHO OBVODU

Syntéza je chápána jako volba a seřízení regulátoru tak, aby byly dosaženy požadavky na regulační pochod. Záleží na struktuře regulátoru (P, I, D) a v nastavení jednotlivých konstant přenosu regulátoru. Při návrhu regulačního obvodu se zatím využívají metody, které nejsou zcela exaktní, ale i tak jsou dobře využitelné. Velmi často se při návrhu využívají vlastní zkušenosti. Syntéza regulačního obvodu vyžaduje znalost vlastností regulované soustavy. Požadavky na regulační pochod se převádějí na matematický model, který nám usnadní samotný návrh. [1], [10], [11]

### 5.1 Struktura regulátoru

Výběr struktury regulátoru je poměrně složité rozhodnutí. Ideální regulátor má velké možnosti nastavení jednotlivých parametrů. Obsluha takového regulátoru vyžaduje větší odbornost. Jednoduchý regulátor obvykle disponuje menšími možnostmi při nastavování jednotlivých parametrů. Důsledkem může být skutečnost, že regulátor nemusí dosáhnout všech požadavků na regulační pochod. V následující tabulce jsou uvedeny základní typy spjitých regulátorů.

Typ regulátoru	Přenos regulátoru
P	$G_R(s) = r_0 = k_p$
I	$G_R(s) = \frac{r_{-1}}{s} = \frac{1}{T_I s}$
PI	$G_R(s) = r_0 + \frac{r_{-1}}{s} = k_p \left( 1 + \frac{1}{T_I s} \right)$
PD	$G_R(s) = r_0 + r_1 s = k_p (1 + T_D s)$
PID	$G_R(s) = r_0 + \frac{r_{-1}}{s} + r_1 s = k_p \left( 1 + \frac{1}{T_I s} + T_D s \right)$

Tabulka 5: Přenosy spjitých regulátorů [9]

Jednoduchý P regulátor v URO má trvalou regulační odchylku při regulaci proporcionálních soustav. Vhodný při malých změnách zatížení, s menším dopravním zpožděním a se střední časovou konstantou.

U I regulátoru v URO dojde k ustálení regulačního pochodu, jakmile regulační odchylka  $e(t)=0$ . Vhodný při pomalých a malých změnách zatížení, bez dopravního zpoždění a s malou časovou konstantou.

Jako jediný nelze použít samostatně D regulátor. Proto je tato složka vždy spojena s P nebo I regulátorem, popřípadě s oběma dohromady. Používá se zejména pro informování regulátoru o změně regulační odchylky.

PI regulátor v URO odstraňuje nevýhody P regulátoru. Je vhodný při velkých a pomalých změnách zatížení, s libovolnými časovými konstantami a s velkým dopravním zpožděním. Odstraňuje trvalou regulační odchylku. Na začátku regulačního pochodu je více patrný vliv proporcionální složky. S přibývajícím časem se projevuje více vliv integrační složky.

PD regulátor vylepšuje díky D složce P regulátor o možnost pracovat s vyšším zesílením regulátoru. PD regulátor zanechává trvalou regulační odchylku, která je však menší než u samotného P regulátoru. Tento typ regulátoru je vhodný při velkém dopravním zpoždění, při malých změnách zatížení a se středními časovými konstantami. Na začátku regulačního pochodu je více patrný vliv derivační složky. S přibývajícím časem se projevuje více vliv proporcionální složky.

PID regulátor v URO se používá ve složitějších případech. I složka odstraňuje trvalou regulační odchylku a vliv D složky zdokonaluje stabilitní vlastnosti regulačního obvodu. Vhodný k regulaci soustav s větším dopravním zpožděním, s libovolnými časovými konstantami, s velkou a rychlou změnou zatížení. Na začátku regulačního pochodu převládá derivační složka a s přibývajícím časem je více patrný vliv integrační složky regulátoru. [1], [7]

## 5.2 Kvalita regulace

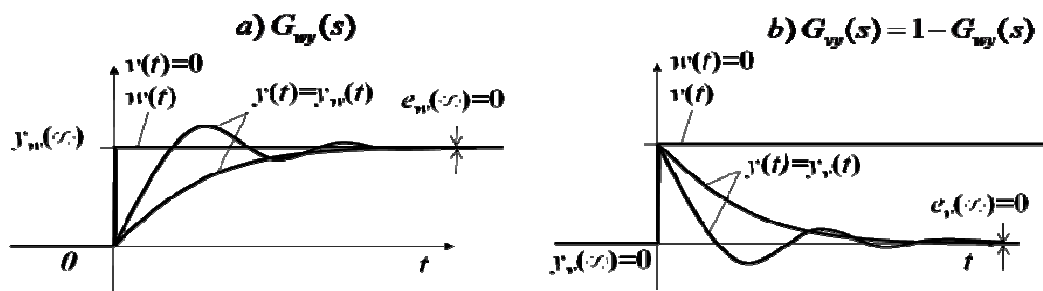
Kvalita regulace je zajištěna správným seřízením regulátoru. Kvalitu regulace můžeme ovlivnit volbou typu regulátoru a nastavením parametrů regulátoru. Kvalita regulace se posuzuje v časové oblasti, kmitočtové oblasti a oblasti komplexní proměnné. [1], [10]

Časová oblast poskytuje rychle a intuitivně zhodnotit kvalitu regulace z průběhu odezvy regulované veličiny  $y(t)$  na skokovou změnu žádané  $w(t)$  nebo poruchové veličiny  $v(t)$ .

$$Y(s) = G_{wy}(s)W(s) + G_{vy}(s)V(s) = Y_w(s) + Y_v(s) \Rightarrow y(t) = y_w(t) + y_v(t) \quad (43)$$

Odezva  $y_w(t)$  je způsobená žádanou veličinou  $w(t)$  při  $v(t)=0$ , odezva  $y_v(t)$  je způsobená poruchovou veličinou  $v(t)$  při  $w(t)=0$ . [6], [10]

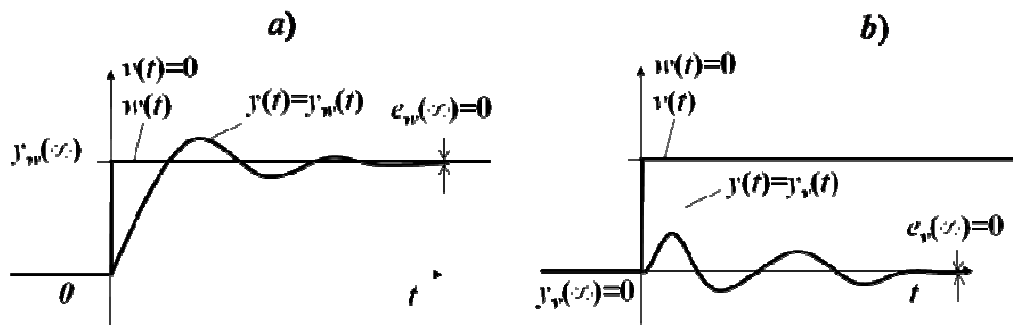
Příklad průběhu kmitavé i aperiodické odezvy regulačního obvodu bez trvalé regulační odchylky jsou na následujícím obrázku. Pokud tedy působí poruchová veličina na výstupu regulované soustavy, tak odezva regulačního obvodu na skokovou změnu poruchové veličiny  $v(t)$  má stejný průběh jako odezva regulačního obvodu na skokovou změnu  $y(t)$  s tím rozdílem, že odezva  $v(t)$  je překllopena podle přímky  $y_w(\infty)$  a posunuta dolů o  $y_w(\infty)$ .



Obrázek 8: Odezvy regulačního obvodu na skokové změny: a) žádané veličiny  $w(t)$ , b) poruchové veličiny  $v(t)$  působící na výstupu regulované soustavy v případě nulových trvalých regulačních odchylek

Nulové trvalé regulační odchylky vzniknou v regulačním obvodu za předpokladu, že obsahuje alespoň jeden integrační člen. Stupněm astatismu rozumíme počet integračních členů v obvodu. [10]

V případě, kdy poruchová veličina  $v(t)$  působí na vstupu regulované soustavy, pak může odezva vypadat následovně:



Obrázek 9: Odezvy regulačního obvodu na skokové změny: a) žádané veličiny  $w(t)$ , b) poruchové veličiny  $v(t)$  působící na vstupu proporcionalní regulované soustavy v případě regulátoru s integrační složkou

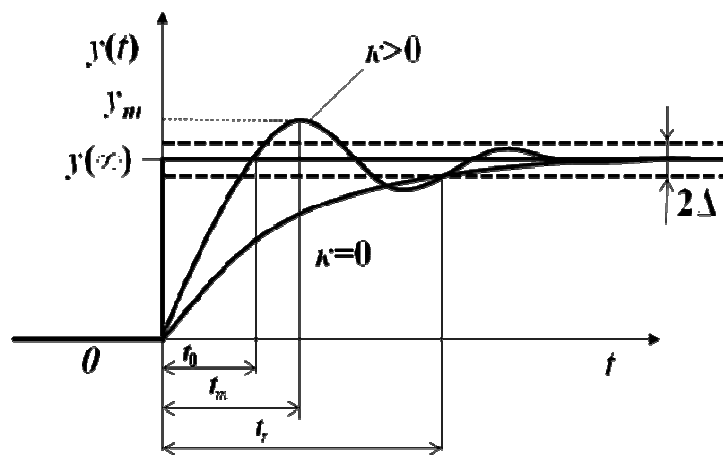
Nulových trvalých regulačních odchylek  $e_w(\infty) = 0$  a  $e_v(\infty) = 0$  se tedy dosáhne použitím regulátoru s integrační složkou. Trvalé regulační odchylky jsou tedy dány následujícími vztahy:

$$E(s) = G_{we}(s)W(s) + G_{ve}(s)V(s) = E_w(s) + E_v(s) \quad (44)$$

$$e_w(\infty) = \lim_{s \rightarrow 0} sE_w(s), \quad e_v(\infty) = \lim_{s \rightarrow 0} sE_v(s)$$

Pokud má přenos poruchy  $G_{vy}(s)$  nebo odchylkový přenos poruchy  $G_{ve}(s)$  derivační charakter, tak je trvalá regulační odchylka způsobená poruchovou veličinou  $v(t)$  nulová.

Při určování kvality regulace jsou nejdůležitějšími parametry doba regulace  $t_r$  a relativní překmit.



Obrázek 10: Přechodové charakteristiky kmitavého a nekmitavého přenosu s ukazateli kvality

Relativní překmit je dán vztahem:

$$\kappa = \frac{y_m - y(\infty)}{y(\infty)}, \quad y_m = y(t_m) \quad (45)$$

Maximální hodnoty regulované veličiny  $y_m$  při překmitu je dosaženo v čase  $t_m$ . Regulovaná veličina se pak ustálí na hodnotě  $y(\infty)$ . Doba regulace  $t_r$  je dána časem, kdy do pásma  $(y(\infty) \pm \Delta)$  o velikosti  $2\Delta$  trvale vejde regulovaná veličina  $y(t)$ . Tolerance regulace je dána rovnicí:

$$\Delta = \delta y(\infty), \quad \delta = 0,01 \div 0,05 \quad (1 \div 5)\% \quad (46)$$

Nejčastěji se používají hodnoty relativní tolerance regulace  $\delta = 0,05$  nebo  $\delta = 0,02$ .

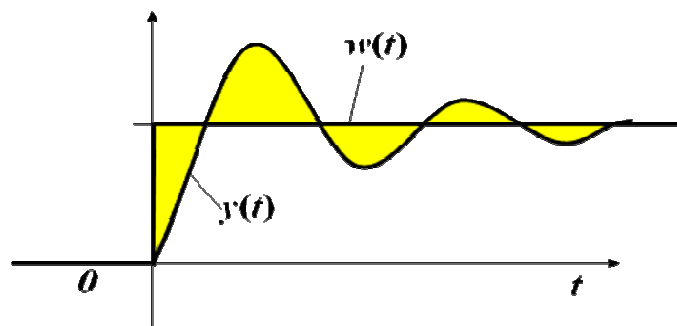
Některé technologické procesy vyžadují, aby byl překmit  $\kappa = 0$ . V takovém případě se jedná o aperiodický systém. U takovýchto systémů se většinou také požaduje, aby regulace proběhla za co možná nejkratší dobu  $t_r$ . [6], [10]

U kmitavých regulačních pochodů je překmit  $\kappa > 0$ . Tyto systémy mají kratší rychlost odezvy  $t_0$ . Rychlostí odezvy se rozumí čas, kdy regulovaná veličina  $y(t)$  poprvé dosáhne hodnoty ustálené hodnoty  $y(\infty)$  nebo čas od dosáhnutí  $0,1y(\infty)$  do dosáhnutí  $0,9y(\infty)$ . [10]

### 5.3 Integrální kritéria

Integrální kritéria se používají pro vyjádření kvality regulačního pochodu. Využívá se tzv. regulační plochy. Regule je tím kvalitnější, čím je menší regulační plocha, která je znázorněna žlutou plochou na obrázku Obrázek 11: Příklad regulační plochy. Minimální regulační plocha vyjadřuje alternativní cestu mezi požadavky na minimální překmit  $\kappa$  a minimální dobu regulace  $t_r$ . Oba dva tyto požadavky si vzájemně odporují. Pokud bude minimální doba regulace  $t_r$ , tak bude velký překmit  $\kappa$ , a naopak pokud bude minimální překmit  $\kappa$ , tak bude zase dlouhá doba regulace  $t_r$ . Důvodem minimální regulační plochy je skutečnost, že při regulačním pochodu dochází k výměně energií. Při podregulování (záporná regulační odchylka) potřebuje regulovaná soustava energii. A při přeregulování (kladná regulační odchylka) naopak regulovaná soustava má energie přebytek a tak předává energii okolí. Ideální by samozřejmě byla skoková změna, což je ovšem nereálné. A proto je cílem, aby byla výměna energií co nejmenší. [1], [6], [10]

Při výpočtech se využívá regulační odchylky  $e(t) = w(t) - y(t)$  a předpokládá se  $e(\infty) = e_w(\infty) = 0$ . V případě, že  $e(\infty) \neq 0$ , tak se místo  $e(t)$  dosazuje vztah  $e(t) - e(\infty)$ .



Obrázek 11: Příklad regulační plochy

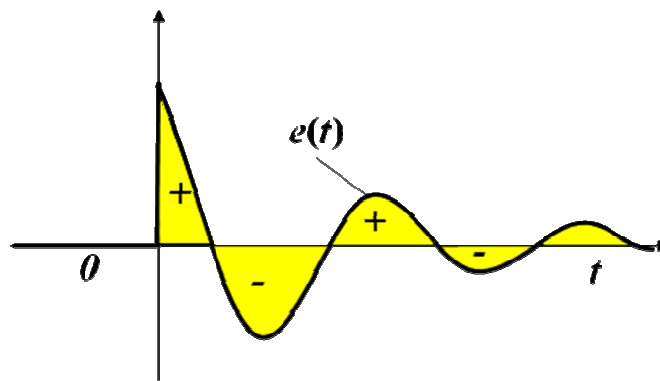
#### 5.3.1 Lineární regulační plocha

Jedná se o nejjednodušší kritérium, které lze snadno spočítat. Označuje se jako  $I_{IE}$  (IE = Integral of Error). Nehodí se pro kmitavé regulační pochody, protože by se plochy vzájemně odečetli a  $I_{IE} = 0$ . [1], [10]

Lineární regulační plocha se vypočítá podle vztahu:

$$I_{IE} = \int_0^{\infty} e(t) dt \quad (47)$$

Lineární regulační plocha pro kmitavé systémy může vypadat například následovně:



Obrázek 12: Lineární regulační plocha

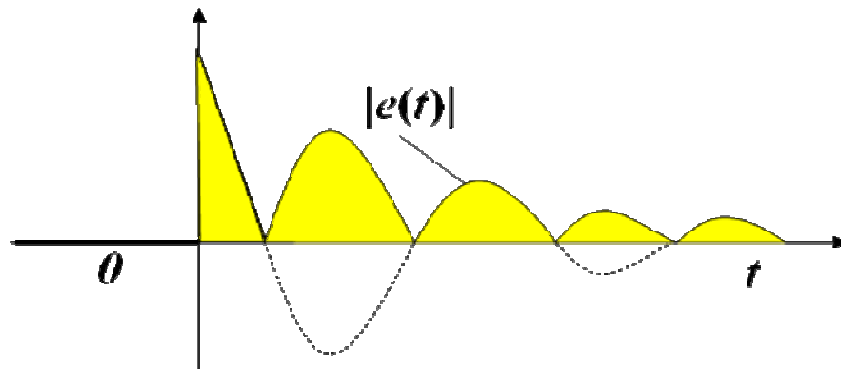
### 5.3.2 Absolutní regulační plocha

Kritérium absolutní regulační plochy  $I_{IAE}$  (IAE = Integral of Absolute Error) díky přidané absolutní hodnotě odstraňuje nevýhodu kritéria lineární regulační plochy. Z tohoto důvodu je toto kritérium vhodné i pro kmitavé systémy. Avšak zásadní nevýhodou tohoto kritéria je neexistující derivace v bodech, ve kterých dochází ke změně znaménka  $e(t)$ . Nelze tedy určit regulační plochu analyticky, ale pouze pomocí simulace. [1], [10]

Absolutní regulační plocha je dána vztahem:

$$I_{IAE} = \int_0^{\infty} |e(t)| dt \quad (48)$$

Absolutní regulační plocha může vypadat následovně:



Obrázek 13: Absolutní regulační plocha

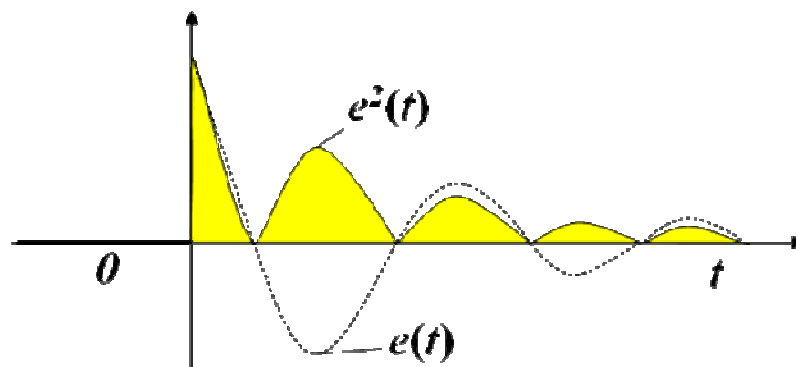
### 5.3.3 Kvadratická regulační plocha

Kritérium se označuje jako  $I_{ISE}$  (ISE = Integral of Squared Error). Díky druhé mocnině odstraňuje nevýhody lineární a absolutní regulační plochy. Regulační plochu lze již vypočítat analyticky. Při použití tohoto kritéria je výsledný průběh regulované veličiny příliš kmitavý. [1], [10]

Vztah pro výpočet kvadratické regulační plochy je dán rovnicí:

$$I_{ISE} = \int_0^{\infty} e^2(t) dt \quad (49)$$

Příklad kvadratické regulační plocha je uveden na následujícím obrázku:



Obrázek 14: Kvadratická regulační plocha

Obraz regulační odchylky je dán vztahem:

$$E(s) = \frac{b_{n-1}s^{n-1} + \dots + b_1s + b_0}{a_n s^n + \dots + a_1s + a_0} = \frac{1}{s} [G_w(s) - G_w(0)] \quad (50)$$

Pro stupně  $n = 1, 2, 3$  platí vztahy:

$$n = 1 \quad I_{ISE} = \frac{b_0^2}{2a_0a_1} \quad (51)$$

$$n = 2 \quad I_{ISE} = \frac{a_0b_1^2 + a_2b_0^2}{2a_0a_1a_2}$$

$$n = 3 \quad I_{ISE} = \frac{a_0a_1b_2^2 + a_0a_3(b_1^2 - 2b_0b_2) + a_2a_3b_0^2}{2a_0a_3(a_1a_2 - a_0a_3)}$$

[10]

### 5.3.4 Kritérium ITAE

Velmi oblíbené kritérium  $I_{ITAE}$  (ITAE = Integral of Time multiplied by Absolute Error), které má v sobě obsaženo čas i regulační odchylku, což způsobuje minimalizaci absolutní regulační plochy a doby regulace  $t_r$ . [1], [10]

Kritérium ITAE je dáno vztahem:

$$I_{ITAE} = \int_0^{\infty} t |e(t)| dt \quad (52)$$

Simulacemi byly určeny standardní tvary přenosu řízení, které jsou pro stupeň astatismu  $q = 1$  a stupeň charakteristického mnohočlenu regulačního obvodu  $n = 2, 3$  dány vztahy:

$$n = 2 \quad G_{wy}(s) = \frac{a^2}{s^2 + 1,4as + a^2} \Rightarrow G_o(s) = \frac{a^2}{s^2 + 1,4as} = \frac{a^2}{s(s + 1,4a)}, \quad (53)$$

$$n = 3 \quad G_{wy}(s) = \frac{a^3}{s^3 + 1,75as^2 + 2,15a^2s + a^3} \Rightarrow$$

$$G_o(s) = \frac{a^3}{s^3 + 1,75as^2 + 2,15a^2s} = \frac{a^3}{s(s^2 + 1,75as + 2,15a^2)},$$

[10]

## 5.4 Metoda požadovaného modelu

Dříve se tato metoda nazývala metoda inverze dynamiky. Jedná se o velmi jednoduchou analyticko-experimentální metodu pro seřizování konvenčních regulátorů. Základem této metody je požadovaný model URO, který je ve tvaru:

$$G_{wy}(s) = \frac{Y(s)}{W(s)} = \frac{a}{s + ae^{-T_d s}} e^{-T_d s} \quad (54)$$

Tato metoda nastavuje stupeň astatismu na  $q = 1$ , z čehož plyne, že seřizuje regulátor tak, že je dosaženo nulové trvalé regulační odchylky. Požadovaný překmit  $\kappa$  regulované veličiny je od 0 do 50%.

Přenos regulátoru je pak dán vztahem:

$$G_R(s) = \frac{1}{G_S(s)} \frac{G_{wy}(s)}{1 - G_{wy}(s)} \quad (55)$$

Přenos doporučeného regulátoru:

$$G_R(s) = \frac{a}{sG_S(s)} e^{-T_d s} \quad (56)$$

Závislost zesílení otevřeného regulačního obvodu  $a$  na relativním překmitu  $\kappa$  je odvozena z charakteristického kvazimnohočlenu regulačního obvodu. [1], [10]

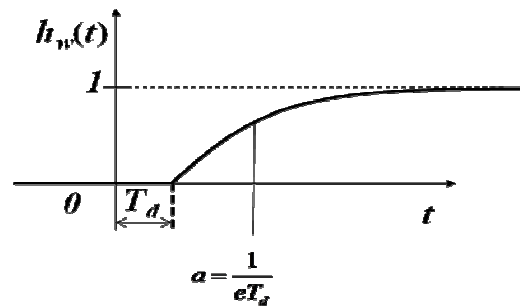
Pro aperiodický regulační pochod platí vztah:

$$a = \frac{1}{eT_d} \quad (57)$$

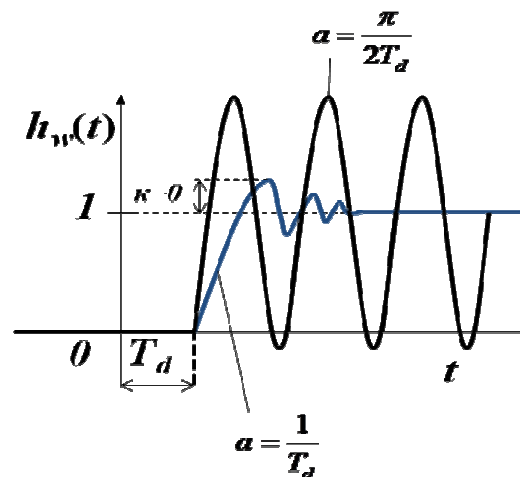
Pro regulační pochod na kmitavé mezi stability platí vztah:

$$a = \frac{\pi}{2T_d} \quad (58)$$

Příklad závislosti zesílení otevřeného regulačního obvodu  $a$  na překmitu  $\kappa$  je zobrazena na následujících obrázcích:



Obrázek 15: Přechodová charakteristika nekmitavého regulačního obvodu



Obrázek 16: Přechodové charakteristiky regulačního obvodu

Obecně pak platí vztah mezi zesílením a relativním překmitem:

$$a = \frac{1}{\beta T_d} \quad (59)$$

Koeficient  $\beta$  byl určen simulačně a lze jej určit z následující tabulky:

$\kappa$	0	0,05	0,10	0,15	0,20	0,25	0,30	0,35	0,40	0,45	0,50
$\alpha$	1,282	0,984	0,884	0,832	0,763	0,697	0,669	0,640	0,618	0,599	0,577
$\beta$	2,718	1,944	1,720	1,561	1,437	1,337	1,248	1,172	1,104	1,045	0,992

Tabulka 6: Závislost koeficientů  $\alpha$  a  $\beta$  na relativním překmitu  $\kappa$  [1], [10]

V následující tabulce jsou popsány vztahy pro výpočet stavitelných parametrů regulátorů.

	Regulovaná soustava $-S$	Regulátor – R				
		Typ	$k_R^*$		$T_I^*$	$T_D^*$
			$T_d=0$	$T_d>0$		
1	$\frac{k_1}{s} e^{-T_d s}$	P	$\frac{(1-c_w)}{k_1 T}$	$\frac{a}{k_1}$	-	-
2		P	$\frac{2}{k_1(2T_w+T)}$	$\frac{a}{k_1}$	-	-
3	$\frac{k_1}{(T_1 s + 1)} e^{-T_d s}$	PI	$\frac{(1-c_w)T_I^*}{k_1 T}$	$\frac{aT_I^*}{k_1}$	$\frac{c_1}{1-c_1} T$	-
4		PI	$\frac{2T_I^*}{k_1(2T_w+T)}$	$\frac{aT_I^*}{k_1}$	$T_1 - 0,5T$	-
5	$\frac{k_1}{s(T_1 s + 1)} e^{-T_d s}$	PD	$\frac{(1-c_w)}{k_1 T}$	$\frac{a}{k_1}$	-	$\frac{c_1}{1-c_1} T$
6		PD	$\frac{2}{k_1(2T_w+T)}$	$\frac{a}{k_1}$	-	$T_1 - 0,5T$
7	$\frac{k_1}{(T_1 s + 1)(T_2 s + 1)} e^{-T_d s}$	PD	$\frac{(1-c_w)T_I^*}{k_1 T}$	$\frac{aT_I^*}{k_1}$	$\frac{c_1 + c_2 - 2c_1 c_2}{1 - c_1 - c_2 + c_1 c_2} T$	$\frac{c_1 c_2 T}{c_1 + c_2 - 2c_1 c_2}$
8		PD	$\frac{2T_I^*}{k_1(2T_w+T)}$	$\frac{aT_I^*}{k_1}$	$T_1 + T_2 - T$	$\frac{4T_1 T_2 - [2(T_1 + T_2) - T]T}{4(T_1 + T_2 - T)}$
9	$\frac{k_1}{T_0^2 s^2 + 2\xi_0 T_0 s + 1} e^{-T_d s}$	PID	$\frac{(1-c_w)T_I^*}{k_1 T}$	$\frac{aT_I^*}{k_1}$	$\frac{2c(b-c)}{1-2bc+c^2} T$	$\frac{c}{2(b-c)} T$
10		PID	$\frac{2T_I^*}{k_1(2T_w+T)}$	$\frac{aT_I^*}{k_1}$	$2\xi_0 T_0 - T$	$\frac{(2T_0 - \xi_0 T)T_0}{4\xi_0 T_0 - 2(1 - \xi_0^2)T}$

Tabulka 7: Hodnoty stavitelných parametrů konvenčních regulátorů pro metodu MPM

V tabulce platí následující vztahy:

$$a = \frac{1}{\alpha T + \beta T_d} \quad (60)$$

$$c_i = e^{-\frac{T}{T_i}}; i = 1, 2, w \quad c = e^{-\frac{\xi_0 T}{T_0}} \quad b = \cos \frac{\sqrt{1 - \xi_0^2}}{T_0} T$$

Dále platí, že řádek č. 9 je pouze pro koeficienty, které splňují podmínku  $b > c$ . V sudých řádcích tabulky jsou uvedeny přibližné vztahy.  $T$  je vzorkovací perioda, která je u analogových regulátorů  $T = 0$  a u číslicových  $T > 0$ .  $T_w$  je časová konstanta URO ( $T_d = 0$ ). Časové konstanty regulované soustavy jsou označeny jako  $T_0$ ,  $T_1$ ,  $T_2$ . Koeficient zesílení je označen jako  $k_I$ ,  $\xi_0$  je koeficient poměrného tlumení regulované soustavy,  $s$  je komplexní proměnná,  $\kappa$  vyjadřuje překmit,  $T_I^*$  je integrační časová konstanta regulátoru,  $T_D^*$  zastupuje derivační časovou konstantu regulátoru a  $k_R^*$  vyjadřuje doporučené zesílení regulátoru.

Postup při seřizování regulátoru je velmi jednoduchý. Nejdříve se přenos regulované soustavy upraví na jeden ze tvarů v tabulce Tabulka 7: Hodnoty stavitelných parametrů konvenčních regulátorů pro metodu MPM. Dalším krokem je určení koeficientu  $\beta$  pro požadovaný překmit  $\kappa$  z tabulky Tabulka 6: Závislost koeficientů  $\alpha$  a  $\beta$  na relativním překmitu  $\kappa$ . Nakonec se dopočítají stavitelné parametry doporučeného regulátoru opět podle tabulky Tabulka 7: Hodnoty stavitelných parametrů konvenčních regulátorů pro metodu MPM. [1], [9], [10]

## 6 PROGRAMOVACÍ JAZYK JAVA

Ostatní vysokoúrovňové programovací jazyky jako například C++ musejí být nejdříve zkompileovány. Teprve poté se dají spustit. Tento nedostatek se pokusila odstranit firma Sun Microsystems. Jejím cílem bylo vytvořit programovací jazyk, který by umožňoval psát programy pro různá zařízení spotřební elektroniky (TV, PC,...). Počátky Javy sahají do roku 1991, kdy byl spuštěn tzv. Zelený projekt. Programátoři nazvali nový programovací jazyk Oak. Avšak záhy přišli na to, že tento programovací jazyk již existuje, a tak jej přejmenovali na programovací jazyk Java. Ve spotřební elektronice se Java příliš neprosadila, ale našla uplatnění u služby WWW. Java se totiž dá spustit na téměř jakémkoli druhu počítače bez nutnosti kompilace. Oficiálně byla Java představena v roce 1995. Postupem času vzniklo několik verzí Javy. Nejrozšířenější je Java 2 Standard Edition (J2SE), dále jsou to edice jako Java 2 Enterprise Edition (J2EE), Java 2 Micro Edition (J2ME).

### 6.1 Základy programovacího jazyku Java

Program v Javě je klasická posloupnost instrukcí jako v každém jiném programovacím jazyce. Počítačovou aplikací se pak rozumí množina programů, které spolu souvisejí a komunikují.

Základem programu v Javě je hlavní třída. Program může obsahovat i další rozšiřující třídy. Programy se dají editovat pomocí běžného textového editoru například poznámkového bloku. Většinou se však využívají vývojová prostředí, která usnadňují úpravu zdrojového kódu programu. Zdrojový kód má koncovku souboru .java. Jak již bylo zmíněno ostatní programovací jazyky kompilují zdrojový kód do objektového kódu, což Java nedělá. Zdrojový kód v Javě je zkompileován do bajtového kódu s příponou .class. Tento bajtový kód je poté spuštěn ve virtuální stroji Javy (JVM), který přeloží bajtový kód do strojového kódu, jež je spuštěn na počítači. Díky JVM lze spustit bajtový kód na jakémkoli typu počítače. [3], [5]

Typický základní program v Javě může vypadat následovně:

```
1 class AhojSvete {  
2 public static void main(String arg[]){  
3 System.out.println("Ahoj světe!");}}
```

Zdrojový kód uvedený výše definuje třídu AhojSvete. Tato třída obsahuje jedinou hlavní metodu main(), která musí být obsažena v každé třídě a je uzavřena do těla pomocí složených závorek. Každá metoda musí mít název metody, parametry metody, tělo metody a návratovou hodnotu metody. V tělo metody může být posloupnost příkazů. Ve zdrojovém kódu, uvedeném výše, má metoda main() ve svém těle jediný příkaz println(), která vypíše na standardní výstup zadaný řetězec. V Javě je každý příkaz ukončen středníkem na konci řádku. V opačném případě zahlásí kompilátor chybu. [3], [5]

### 6.1.1 Datové typy proměnných

Proměnná je místo v paměti počítače, které je rezervováno pro uchování příslušných dat pro program. Následující tabulka zobrazuje základní datové typy v jazyku Java.

Datový typ	Velikost datového typu [bit]	Slovní popis
byte	8	celé číslo s rozsahem 255(byte)
short	16	“malé“ celé číslo
int	32	“střední“ celé číslo
long	64	“velké“ celé číslo
char	16	znak
float	32	desetinné číslo
double	64	“velké“ desetinné číslo
boolean	1	booleovská hodnota

Tabulka 8: Základní datové typy v jazyku Java

Proměnná se deklaruje podobně jako v ostatních programovacích jazycích. A to například tímto způsobem:

```
1 int CeleCislo;
```

### 6.1.2 Řídící příkazy

Řídící příkazy určují pořadí vykonávání příkazů ve zdrojovém kódu. Mohou to být výběrové příkazy, iterační příkazy nebo skokové příkazy.

Příkladem výběrového řídicího příkazu je podmínka `if...else`. Pokud výraz (typu boolean) za podmínkou `if` platí, tak se provede tělo `if`. Pokud výraz neplatí, tak se provede tělo `else`. Dalším typem výběrového příkazu je přepínač `switch`, který má jednotlivé větve `case`. Výraz za `switch` se porovná se všemi možnými větvemi `case` a podle toho se určitá větev provede. Pokud není výraz roven žádnému výrazu v `case` větvi, tak se provede blok `default`. [3], [5]

Jednoduchý příklad na použití příkazu `if...else`:

```
1 class Priklad {
2 public static void main(String arg[]){
3 int a=5,b=2;
4 if(a==b){System.out.println("Číslo se rovnají.");}
5 else {System.out.println("Číslo se nerovná.");}
6 }
```

V této třídě se porovnají dvě čísla. Pokud se rovnají, tak je podmínka `if` splněna a provede se tělo za `if`. Což znamená, že se vypíše řetězec "Číslo se rovnají.". Pokud by podmínka za `if` splněna nebyla, tak by se provedlo tělo za klíčovým slovem `else`. Ve zdrojovém kódu, který je zobrazen výše, podmínka za `if` neplatí, takže se vypíše na obrazovku řetězec "Číslo se nerovná..".

Jednoduchý příklad na použití příkazu `switch`, kdy se na obrazovku vypíše řetězec "a = 5".

```
1 class Priklad {
2 public static void main(String arg[]){
3 int a=5;
4 switch(a){
5 case 0:
6 System.out.println("a=0");
7 break;
8 case 5:
9 System.out.println("a=5");
10 break;
11 default:
12 System.out.println("a není rovno 0 ani 5");
13 }
14 }
15 }
```

Dalším typem řídicích příkazů jsou iterační příkazy. Zde se provádějí příkazy v těle daného iteračního příkazu do té doby, dokud je splněna určitá podmínka. Příkladem iteračních příkazů může být for, do while nebo while.

Jednoduchý příklad na použití příkazu for, kdy se vypíší čísla od 1 do 10:

```
1 class Priklad {
2 public static void main(String arg[]){
3 int i;
4 for(i=1;i<=10;i++){
5 System.out.println(i);}
6 }}
```

Tělo cyklu for se bude provádět do té doby dokud bude splněna podmínka. V příkladu je to podmínka  $i \leq 10$ .

Iterační příkaz while se označuje také jako cyklus s podmínkou na začátku. Tělo cyklu while se provádí do té doby, dokud je splněna vstupní podmínka.

Jednoduchý příklad na použití příkazu while, kdy se vypíší čísla od 1 do 10:

```
1 class Priklad {
2 public static void main(String arg[]){
3 int i=1;
4 while(i<=10){
5 System.out.println(i);
6 i++;}
7 }}
```

Iterační příkaz do while se také nazývá jako cyklus s podmínkou na konci. Tělo cyklu do while se provede vždy minimálně jednou. Pak se vyhodnotí podmínka na konci a pokud platí, tak se přesune řízení programu opět na začátek cyklu do while.

Jednoduchý příklad na použití příkazu do while, kdy se vypíší čísla od 1 do 10:

```
1 class Priklad {
2 public static void main(String arg[]){
3 int i=1;
4 do{
6 System.out.println(i);
7 i++;} while(i<=10); }}
```

Posledním typem řídicích příkazů jsou skokové příkazy, které přesměrují řízení programu do jiné části. Příkladem je příkaz `return`, který se umísťuje do jednotlivých metod a vrátí provádění programu zpět na místo, odkud byla metoda zavolána. Další skokový příkaz je příkaz `break`, který ukončí aktuální smyčku a pokračuje dál za ní. Naproti tomu skokový příkaz `continue`, také ukončí běh aktuální smyčky, ale řízení programu se přesouvá na začátek smyčky. [3], [5]

### 6.1.3 Výjimky

Výjimkou se rozumí stav programu, který by nastat neměl. Program v tomto stavu neví, co má dělat. Proto by se měly výjimky ošetřovat takovým způsobem, aby se zabránilo zatuhnutí programu. Existují dva druhy výjimek. Výjimka může nastat při kompilaci nebo za běhu programu. Pokud nastane výjimka při kompilaci, tak se nejčastěji jedná o nějakou syntaktickou chybu, kterou vytvořil programátor. Pokud nastane výjimka za běhu, tak se stalo něco neočekávaného. Většinou se jedná o dělení nulou, nekonečný cyklus, špatně napsanou podmínku, která nikdy nenastane atd. Výjimky se ošetřují pomocí příkazů `try` a `catch`. Do těla příkazu `try` se vloží část zdrojového kódu, kde můžeme očekávat výjimku. Tělo `catch` pak obsluhuje ošetření dané výjimky. [3], [5]

Jednoduchý příklad na ošetření výjimek, kdy se program pokouší dělit nulou:

```
1 class Priklad {
2     public static void main(String arg[]){
3         int a=1,b=0,c;
4         try{     c=a/b; }
5         catch(Exception e){
6             System.out.println("Dělení nulou. Vygenerována výjimka "+e);}
7     }
8 }
```

### 6.1.4 Vlákna

Vlákna slouží k paralelnímu zpracování dvou a více úloh v jednom okamžiku. Přepínání mezi jednotlivými vlákny se nazývá přepínání kontextu, o které se stará režie. Vlákna se mohou nacházet ve čtyřech různých stavech (běžící, pozastavené, blokováno, ukončené). Běžící vlákno se právě provádí. Pozastavené vlákno je zastaveno a čeká na obnovení. Blokováno vlákno čeká na zdroj, který používá jiné vlákno. Ukončené vlákno je zastaveno. Každé vlákno má svou prioritu, podle které se určuje důležitost daného vlákna. [3], [5]

Vlákna se v Javě vytvářejí ze třídy Thread a rozhraní Runnable. Každý program obsahuje alespoň jedno vlákno. Toto vlákno se nazývá hlavní vlákno a je vytvořeno automaticky.

Nejčastější metody při používání vláken jsou následující:

Název metody	Popis metody
getName()	Vrací název vlákna.
getPriority()	Vrací prioritu vlákna.
isAlive()	Vrací, zda vlákno běží.
Join()	Pozastavení provádění až do ukončení vlákna.
Run()	Vstupní bod vlákna.
Sleep()	Pozastavení vlákna.
Start()	Spuštění vlákna.

Tabulka 9: Nejčastěji používané metody vláken

Vlastní vlákno se vytváří implementací rozhraní Runnable. Třída, která má implementováno rozhraní Runnable musí mít definovanou metodu run(), aby bylo možné toto rozhraní použít. Metoda run() musí být veřejná (public). Tato metoda obsahuje tělo nového vlákna a vykoná se při jeho spuštění. Dále musí být definován konstruktor nového vlákna, a to ve tvaru Thread (Runnable trida\_s\_implementation\_runnable, String nazev\_vlakna). Z tabulky výše je zřejmé, že nové vlákno se spustí metodou start(). Po dokončení metody run() se vrací řízení programu zpět na místo, odkud byla metoda run() zavolána. [3], [5]

Jednoduchý příklad na práci s vlákny:

```

1 class PrikladVlakno implements Runnable {
2     Thread vlakno;
3     PrikladVlakno(){
4         vlakno=new Thread(this,"Příklad vlákna");
5         vlakno.start();}
6     public void run(){
7         System.out.println("Vedlejší vlákno start");
8         System.out.println("Vedlejší vlákno stop");}
9     }
10 class Hlavni{

```

```
11 public static void main(String arg[]){
12     new PrikladVlakno();
13     System.out.println("Hlavní vlákno start");
14     System.out.println("Hlavní vlákno stop");}
15 }
```

Jako první se na obrazovku vypíše řetězec “Vedlejší vlákno start“, následuje výpis řetězce “Hlavní vlákno start“. Dále se zobrazí řetězec “Vedlejší vlákno stop“ a posledním zobrazeným řetězcem je “Hlavní vlákno stop“. [3], [5]

### 6.1.5 GUI

Tímto termínem se označuje grafické uživatelské rozhraní. GUI se využívá pro propojení s uživatelem. Díky přehlednému GUI se uživatel v příslušné aplikaci neztratí a ví, jak program ovládat. Pro uživatele je mnohem jednodušší ovládat program pomocí grafických prvků, než psát složité příkazy.

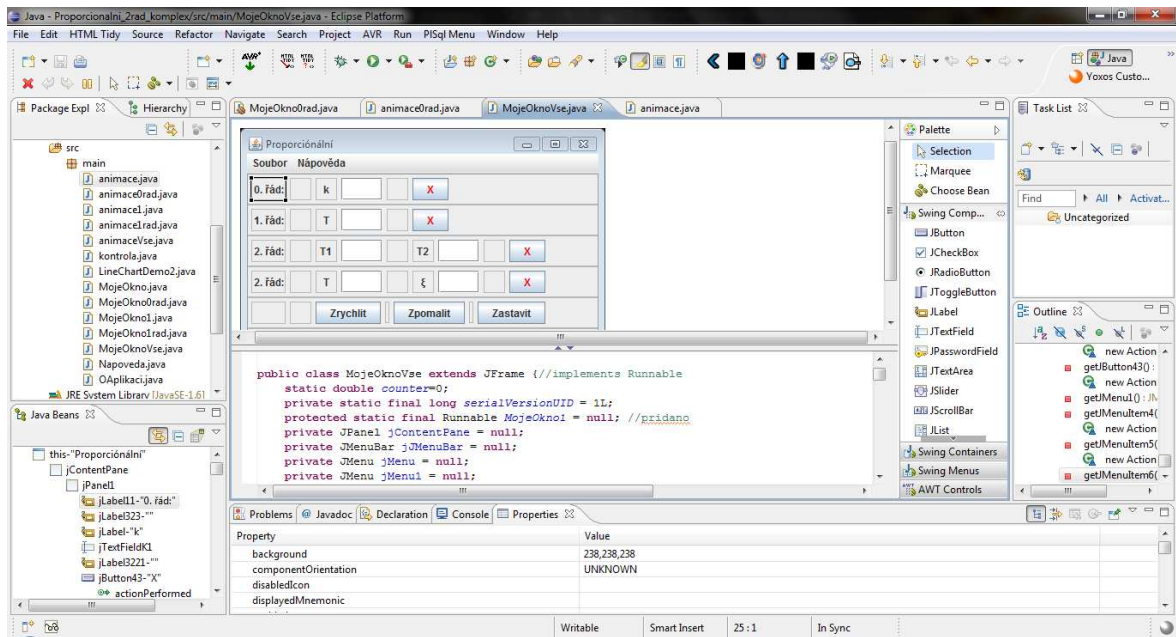
V Javě se pro vytváření GUI často používají balíčky `java.swing` a `java.awt`, které obsahují spoustu grafických tříd a příslušných metod, kterými lze jednotlivé grafické prvky nastavit. [3], [5]

## 6.2 Vývojové prostředí Eclipse 3.5

Jak již bylo zmíněno zdrojový kód v Javě se dá napsat i v běžném poznámkovém bloku. Avšak mnohem komfortnější cestou je využití některého z vývojových prostředí. Mezi známé a zdarma dostupné vývojové prostředí patří například Netbeans nebo Eclipse.

Vývojové prostředí výrazně ulehčuje vytváření, úpravu a správu vytvořených aplikací. Grafické prostředí zobrazuje číslování řádků kódu, barevně odlišuje klíčová slova, upozorňuje na syntaktické chyby, zobrazuje nápovědu na řešení chyb, atd. Také vytváření GUI rozhraní je mnohem jednodušší.

Vývojové prostředí Eclipse 3.5 vytváří prázdné okno při založení nové vizuální třídy `Visual Class`. Velkou výhodou je možnost využít funkci `drag&drop` pro vkládání grafických prvků do implicitně vytvořeného okna. Programátor tak může velmi jednoduše pomocí vývojového prostředí přetahovat jeho požadované grafické prvky přímo do okna. Po přetáhnutí se automaticky vygeneruje zdrojový kód, který se při každé změně aktualizuje. Grafické prvky je také možno editovat přímo ve zdrojovém kódu.



Obrázek 17: Ukázka vývojového prostředí Eclipse 3.5

## **II. PRAKTICKÁ ČÁST**

## 7 LOGICKÉ FUNKCE

Jako první byla vytvořena doplňková prezentace na téma logické funkce. V této prezentaci jsou popsány a vysvětleny základní logické funkce (OR, AND, NOR, NAND, NON, XOR), kanonické tvary, základní pravidla Booleovy algebry, Karnaughova mapa a Quineova-McCluskeyova metoda.

### 7.1 Kanonické tvary – řešený příklad

Úkolem je určit součtový a součinnový tvar logické funkce zadané tabulkou:

$x_1$	$x_2$	$y$
0	0	0
0	1	1
1	0	1
1	1	0

Tabulka 10: Logická funkce – pravdivostní tabulka

Součtový tvar se určí tak, že se v pravdivostní tabulce vyhledají řádky, kde  $y = 1$ . Proměnné, které se rovnají 0 se znegují. Proměnné v daném řádku se pak zapíše jako součin. Jednotlivé součiny proměnných se následně sečtou. Výsledkem je pak součet součinů ve tvaru:

$$y = \overline{x_1}x_2 + x_1\overline{x_2} \quad (61)$$

Součinnový tvar se určí úplně stejným způsobem jako tvar součinnový. S tím rozdílem, že se zapisují řádky, ve kterých je  $y = 0$ . Dalším rozdílem je zápis proměnných v daném řádku jako součet. Tyto součty se pak dají do součinu. Výsledek je pak součinem součtů:

$$y = (x_1 + x_2)(\overline{x_1} + \overline{x_2}) \quad (62)$$

## 7.2 Karnaughova mapa – řešený příklad

Cílem je minimalizovat logickou funkci zadanou v součtovém tvaru:

$$y = \overline{x_1}x_2x_3 + x_1\overline{x_2}x_3 + x_1x_2\overline{x_3} + x_1x_2x_3 \quad (63)$$

Z rovnice je vidět, že funkce má 3 vstupní proměnné  $x_1x_2x_3$ . Tabulka bude mít  $2^3=8$  buněk. Pokud je vstup negovaný, tak jeho souřadnice odpovídá v tabulce logické nule. Pokud vstup není negovaný, tak jeho souřadnice odpovídá logické jedničce. Protože je funkce zadána v součtovém tvaru, tak jednotlivé součiny odpovídají kombinacím, kdy je výstup roven jedničce. Tyto jedničky jsou zapsány do tabulky a následné dvojice jedniček jsou seskupeny.

	$x_2, x_3$			
$x_1$	00	01	11	10
0			1	
1	1		1	1

Obrázek 18: Karnaughova mapa

Proměnné seskupených dvojic (mohou to být i čtveřice, osmice,...), které nemění svou souřadnici řádku ani sloupce, se zapisují do výsledného tvaru. U vstupní proměnné, kde se mění souřadnice, tak se nezapisuje tato vstupní proměnná do výsledného tvaru. Výsledkem je pak tvar:

$$y = x_1\overline{x_3} + x_2x_3 \quad (64)$$

## 7.3 Quineova-McCluskeyova metoda – řešený příklad

Tato metoda má podobný princip jako Karnaughova mapa. Je zadána funkce pro minimalizaci ve tvaru:

$$X(A, B, C) = ABC\overline{C} + \overline{A}BC + \overline{A}BC + ABC \quad (65)$$

Důležité je, aby všechny členy v součtovém tvaru obsahovaly všechny vstupní proměnné. Poté se zapíšou hodnoty jednotlivých součinů do tabulky. V tabulce je výhodné seskupit součiny podle počtu jedniček, které obsahují. Pak se seskupují dvojice, které se liší maximálně v jednom znaku. Na místě, kde se liší, je vložena pomlčka.

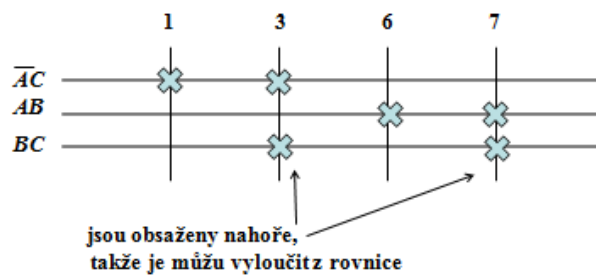
1	001	1,3	0-1
3	011	3,7	-11
6	110	6,7	11-
7	111		

Obrázek 19: Tabulka seskupených dvojic

Jakmile už nelze krátit, tak se seskupené dvojice sepíší do výsledné funkce.

$$X(A, B, C) = \overline{A}C + BC + AB \quad (66)$$

Z výsledné funkce se vytvoří mřížka prostých implikantů.



Obrázek 20: Mřížka prostých implikantů

Jak je vidět z obrázku výše, dvojici  $BC$  lze vynechat, jelikož jejich hodnoty již jsou obsaženy v jiných členech. Výsledná funkce má pak tvar:

$$X(A, B, C) = \overline{A}C + AB \quad (67)$$

## 8 LAPLACEOVA TRANSFORMACE

Do již existující prezentace 3.Transformace.ppt byly vloženy ilustrativní příklady na téma Laplaceovy transformace. Je znázorněn postup při řešení Laplaceovy transformace pomocí její základní definice, řešení pomocí rozkladu na parciální zlomky a řešení pomocí Heavisideova rozvoje. V posledním řadě je vyřešena jednoduchá diferenciální rovnice pomocí Laplaceovy transformace.

### 8.1 Laplaceova transformace – řešený příklad 1

Úkolem je převést funkci  $f(t)=t$  pomocí základní definice Laplaceovy transformace do obrazu. Řešením je následující postup:

$$L\{t\} = \int_0^{\infty} t \cdot e^{-st} dt = \left[ t \left( -\frac{1}{s} e^{-st} \right) \right]_0^{\infty} - \int_0^{\infty} \left( -\frac{1}{s} e^{-st} \right) dt \quad (68)$$

$$L\{t\} = 0 + \frac{1}{s^2} \left[ -e^{-st} \right]_0^{\infty} = \frac{1}{s^2} [0 - (-1)] = \frac{1}{s^2}$$

### 8.2 Laplaceova transformace – řešený příklad 2

Funkce  $f(t)=e^{-at}$  se má převést bez použití slovníku Laplaceovy transformace do obrazu. Řešení je následující:

$$L\{e^{-at}\} = \int_0^{\infty} e^{-at} \cdot e^{-st} dt = \int_0^{\infty} e^{-(a+s)t} dt = \left[ \frac{e^{-(a+s)t}}{-(a+s)} \right]_0^{\infty} = -\frac{1}{a+s} \left[ e^{-(a+s)t} \right]_0^{\infty} = \quad (69)$$

$$L\{e^{-at}\} = -\frac{1}{a+s} [0 - 1] = \frac{1}{s+a}$$

### 8.3 Laplaceova transformace – řešený příklad 3

Funkce je zadána ve tvaru  $f(t) = \sin \omega t$  a úkolem je určit její obraz pomocí Laplaceovy transformace. Pro zjednodušení je využito Eulerových vztahů:

$$\begin{aligned} e^{j\omega t} &= \cos \omega t + j \sin \omega t \\ \sin \omega t &= \frac{e^{j\omega t} - e^{-j\omega t}}{2j} \\ \cos \omega t &= \frac{e^{j\omega t} + e^{-j\omega t}}{2} \end{aligned} \quad (70)$$

Po dosazení za  $\sin \omega t$  do funkce  $f(t)$ :

$$\begin{aligned} L\{\sin \omega t\} &= \int_0^{\infty} (\sin \omega t) \cdot e^{-st} dt = \int_0^{\infty} \left( \frac{e^{j\omega t} - e^{-j\omega t}}{2j} \right) \cdot e^{-st} dt = \\ &= \frac{1}{2j} \int_0^{\infty} (e^{j\omega t} \cdot e^{-st} - e^{-j\omega t} \cdot e^{-st}) dt = \frac{1}{2j} \int_0^{\infty} e^{-(j\omega+s)t} dt - \frac{1}{2j} \int_0^{\infty} e^{-(j\omega+s)t} dt = \\ &= \frac{1}{2j} \left[ \frac{e^{-(j\omega+s)t}}{-(-j\omega+s)} \right]_0^{\infty} - \frac{1}{2j} \left[ \frac{e^{-(j\omega+s)t}}{-(j\omega+s)} \right]_0^{\infty} = \frac{1}{2j} \left\{ -\frac{1}{s-j\omega} [e^{-(s-j\omega)t}]_0^{\infty} + \frac{1}{s+j\omega} [e^{-(s+j\omega)t}]_0^{\infty} \right\} = \\ &= \frac{1}{2j} \left\{ -\frac{1}{s-j\omega} [0-1] + \frac{1}{s+j\omega} [0-1] \right\} = \frac{1}{2j} \left\{ \frac{1}{s-j\omega} - \frac{1}{s+j\omega} \right\} = \\ &= \frac{1}{2j} \left\{ \frac{(s+j\omega) - (s-j\omega)}{(s-j\omega)(s+j\omega)} \right\} = \frac{1}{2j} \left\{ \frac{s+j\omega - s + j\omega}{s^2 - j^2 \omega^2} \right\} = \frac{1}{2j} \left\{ \frac{2j\omega}{s^2 + \omega^2} \right\} = \underline{\underline{\frac{\omega}{s^2 + \omega^2}}} \end{aligned} \quad (71)$$

### 8.4 Laplaceova transformace – řešený příklad 4

Je zadána funkce ve tvaru  $f(t) = \cos \omega t$  uťo. Podobně jako v předešlém příkladu je využito Eulerových vztahů.

$$\begin{aligned}
 L\{\cos \omega t\} &= \int_0^{\infty} (\cos \omega t) \cdot e^{-st} dt = \int_0^{\infty} \left( \frac{e^{j\omega t} + e^{-j\omega t}}{2} \right) \cdot e^{-st} dt = & (72) \\
 &= \frac{1}{2} \int_0^{\infty} (e^{j\omega t} \cdot e^{-st} + e^{-j\omega t} \cdot e^{-st}) dt = \frac{1}{2} \int_0^{\infty} e^{-(j\omega+s)t} dt + \frac{1}{2} \int_0^{\infty} e^{-(j\omega+s)t} dt = \\
 &= \frac{1}{2} \left[ \frac{e^{-(j\omega+s)t}}{-(-j\omega+s)} \right]_0^{\infty} + \frac{1}{2} \left[ \frac{e^{-(j\omega+s)t}}{-(j\omega+s)} \right]_0^{\infty} = \frac{1}{2} \left\{ -\frac{1}{s-j\omega} \left[ e^{-(s-j\omega)t} \right]_0^{\infty} - \frac{1}{s+j\omega} \left[ e^{-(s+j\omega)t} \right]_0^{\infty} \right\} = \\
 &= \frac{1}{2} \left\{ -\frac{1}{s-j\omega} [0-1] - \frac{1}{s+j\omega} [0-1] \right\} = \frac{1}{2} \left\{ \frac{1}{s-j\omega} + \frac{1}{s+j\omega} \right\} = \\
 &= \frac{1}{2} \left\{ \frac{(s+j\omega) + (s-j\omega)}{(s-j\omega)(s+j\omega)} \right\} = \frac{1}{2} \left\{ \frac{s+j\omega+s-j\omega}{s^2-j^2\omega^2} \right\} = \frac{1}{2} \left\{ \frac{2s}{s^2+\omega^2} \right\} = \underline{\underline{\frac{s}{s^2+\omega^2}}}
 \end{aligned}$$

### 8.5 Laplaceova transformace – řešený příklad 5

Dalším typem příkladu může být určení originálu funkce z daného obrazu. Obraz je zadán ve tvaru:

$$\frac{5s+2}{2s^3+6s^2+4s} \quad (73)$$

Rozkladem na parciální zlomky se dostanou tvary, které již lze nalézt ve slovníku Laplaceovy transformace, který je obsažen v příloze této bakalářské práce. Dosazovací metodou se pak určí koeficienty jednotlivých zlomků.

$$\frac{5s+2}{2s^3+6s^2+4s} = \frac{5s+2}{2s(s^2+3s+2)} = \frac{\frac{1}{2}(5s+2)}{s(s+1)(s+2)} = \frac{A}{s} + \frac{B}{s+1} + \frac{C}{s+2} \quad / \cdot s(s+1)(s+2) \quad (74)$$

$$\frac{1}{2}(5s+2) = A(s+1)(s+2) + B(s+2)s + C(s+1)s$$

$$s=0 \Rightarrow \frac{1}{2}(5 \cdot 0 + 2) = A(0+1)(0+2) + B(0+2)0 + C(0+1)0 \Rightarrow 1 = 2A \Rightarrow A = \frac{1}{2}$$

$$s=-1 \Rightarrow -\frac{3}{2} = B(-1+2)(-1) \Rightarrow -\frac{3}{2} = -B \Rightarrow B = \frac{3}{2}$$

$$s=-2 \Rightarrow -4 = C(-2+1)(-2) \Rightarrow -4 = 2C \Rightarrow C = -2$$

$$\frac{5s+2}{2s^3+6s^2+4s} = \frac{\frac{1}{2}}{s} + \frac{\frac{3}{2}}{s+1} + \frac{-2}{s+2} \Rightarrow \underline{\underline{f(t) = \frac{1}{2} + \frac{3}{2}e^{-t} - 2e^{-2t}}}}$$

Místo metody rozkladu na parciální zlomky, lze využít i Heavisideova rozvoje:

$$\frac{2,5s+1}{s(s+1)(s+2)} = \frac{A}{s} + \frac{B}{s+1} + \frac{C}{s+2} \quad (75)$$

$$A = \left[ s \frac{2,5s+1}{s(s+1)(s+2)} \right]_{s=0} = \left[ \frac{2,5s+1}{(s+1)(s+2)} \right]_{s=0} = \frac{1}{2}$$

$$B = \left[ \frac{2,5s+1}{s(s+2)} \right]_{s=-1} = \frac{3}{2}$$

$$C = \left[ \frac{2,5s+1}{s(s+1)} \right]_{s=-2} = -2$$

$$\frac{2,5s+1}{s(s+1)(s+2)} = \frac{1}{2} \frac{1}{s} + \frac{3}{2} \frac{1}{s+1} - 2 \frac{1}{s+2}$$

$$\underline{\underline{f(t) = \frac{1}{2} + \frac{3}{2}e^{-t} - 2e^{-2t}}}}$$

## 8.6 Laplaceova transformace – řešený příklad 6

Obraz funkce může být zadán i ve tvaru, kdy má jmenovatel obrazu vícenásobný kořen.

$$\frac{3s+4}{(s+1)(s+2)^2} \quad (76)$$

Řešení je opět dosaženo rozkladem na parciální zlomky. Opět může být využita dosazovací metoda pro určení koeficientů zlomků.

$$\begin{aligned} \frac{3s+4}{(s+1)(s+2)^2} &= \frac{A}{s+1} + \frac{B}{s+2} + \frac{C}{(s+2)^2} \quad / \cdot (s+1)(s+2)^2 & (77) \\ 3s+4 &= A(s+2)^2 + B(s+2)(s+1) + C(s+1) \\ s = -2 &\Rightarrow -6+4 = -C \Rightarrow C = 2 \\ s = -1 &\Rightarrow 1 = A \\ s = 0 &\Rightarrow 4 = 4A + 2B + C \Rightarrow 4 = 4 \cdot 1 + 2B + 2 \Rightarrow B = -1 \\ \frac{3s+4}{(s+1)(s+2)^2} &= \frac{1}{s+1} + \frac{-1}{s+2} + \frac{2}{(s+2)^2} \Rightarrow \underline{\underline{f(t) = e^{-t} - e^{-2t} + 2te^{-2t}}} \end{aligned}$$

Avšak lze také využít metodu srovnání koeficientů, popřípadě obě tyto metody kombinovat.

$$\begin{aligned} \frac{3s+4}{(s+1)(s+2)^2} &= \frac{A}{s+1} + \frac{B}{s+2} + \frac{C}{(s+2)^2} \quad / \cdot (s+1)(s+2)^2 & (78) \\ 3s+4 &= A(s+2)^2 + B(s+2)(s+1) + C(s+1) \\ 3s+4 &= As^2 + 4sA + 4A + Bs^2 + 3sB + 2B + Cs + C \\ s^2 : &\Rightarrow 0 = A + B \Rightarrow A = -B \Rightarrow A = 1 \\ s^1 : &\Rightarrow 3 = 4A + 3B + C \Rightarrow 3 = -4B + 3B + C \Rightarrow C = 3 + B \Rightarrow C = 2 \\ s^0 : &\Rightarrow 4 = 4A + 2B + C \Rightarrow 4 = -4B + 2B + 3 + B \Rightarrow B = -1 \\ \frac{3s+4}{(s+1)(s+2)^2} &= \frac{1}{s+1} + \frac{-1}{s+2} + \frac{2}{(s+2)^2} \Rightarrow \underline{\underline{f(t) = e^{-t} - e^{-2t} + 2te^{-2t}}} \end{aligned}$$

Další možností je využití Heavisideova rozvoje:

$$\frac{3s+4}{(s+1)(s+2)^2} = \frac{A}{s+1} + \frac{B}{s+2} + \frac{C}{(s+2)^2} \quad (79)$$

$$A = \left[ (s+1) \frac{3s+4}{(s+1)(s+2)^2} \right]_{s=-1} = \left[ \frac{3s+4}{(s+2)^2} \right]_{s=-1} = 1$$

$$C = \left[ (s+2)^2 \frac{3s+4}{(s+1)(s+2)^2} \right]_{s=-2} = \left[ \frac{3s+4}{s+1} \right]_{s=-2} = 2$$

$$B = \frac{1}{(2-1)!} \left\{ \frac{d^{(2-1)}}{ds^{(2-1)}} \left[ (s+2)^2 \frac{3s+4}{(s+1)(s+2)^2} \right] \right\}_{s=-2} = \frac{1}{1!} \left\{ \frac{d}{ds} \left[ \frac{3s+4}{s+1} \right] \right\}_{s=-2}$$

$$B = \left\{ \frac{3}{s+1} - \frac{3s+4}{(s+1)^2} \right\}_{s=-2} = -3 + 2 = -1$$

$$\frac{3s+4}{(s+1)(s+2)^2} = \frac{1}{s+1} - \frac{1}{s+2} + 2 \frac{1}{(s+2)^2}$$

$$\underline{\underline{f(t) = e^{-t} - e^{-2t} + 2te^{-2t}}}$$

### 8.7 Laplaceova transformace – řešený příklad 7

Jmenovatel obrazu funkce může mít také komplexně sdružené kořeny.

$$\frac{5}{s^2 + 2s + 3} \quad (80)$$

I v tomto případě se zlomek rozloží na parciální zlomky, které jsou však v trochu odlišném tvaru.

$$\begin{aligned} \frac{5}{s^2 + 2s + 3} &= \frac{As + B}{s^2 + 2s + 3} = \frac{As + B}{(s+1)^2 + 2} = \frac{\sqrt{2}C}{(s+1)^2 + 2} + \frac{(s+1)D}{(s+1)^2 + 2} \quad / \cdot ((s+1)^2 + 2) \quad (81) \\ 5 &= \sqrt{2}C + (s+1)D \\ s = -1 &\Rightarrow 5 = \sqrt{2}C \Rightarrow C = \frac{5}{\sqrt{2}} = \frac{5\sqrt{2}}{2} \\ s = 0 &\Rightarrow 5 = \sqrt{2}C + D \Rightarrow 5 = \sqrt{2} \frac{5\sqrt{2}}{2} + D \Rightarrow D = 0 \\ \frac{5}{s^2 + 2s + 3} &= \frac{\sqrt{2} \frac{5\sqrt{2}}{2}}{(s+1)^2 + 2} + \frac{(s+1)0}{(s+1)^2 + 2} \Rightarrow \underline{\underline{f(t) = \frac{5\sqrt{2}}{2} e^{-t} \sin \sqrt{2}t}} \end{aligned}$$

### 8.8 Laplaceova transformace – řešený příklad 8

Laplaceovu transformaci lze také využít k řešení diferenciálních rovnic. Je zadána diferenciální rovnice ve tvaru:

$$y''(t) + \omega^2 y(t) = 0 \quad \omega > 0, y(0) = 0, y'(0) = 5 \quad (82)$$

Převedením do obrazu pomocí Laplaceovy transformace je získání řešení značně jednodušší:

$$\begin{aligned}
 y''(t) + \omega^2 y(t) &= 0 & (83) \\
 s^2 Y(s) - sy(0) - y'(0) + \omega^2 Y(s) &= 0 \\
 Y(s)(s^2 + \omega^2) - 5 &= 0 \\
 Y(s) &= \frac{5}{s^2 + \omega^2} = \frac{5}{s^2 + \omega^2} \frac{\omega}{\omega} = \frac{5}{\omega} \frac{\omega}{s^2 + \omega^2} \\
 y(t) &= \underline{\underline{\frac{5}{\omega} \sin \omega t}}
 \end{aligned}$$

## 8.9 Laplaceova transformace – řešený příklad 9

Úkolem je vyřešit zadanou diferenciální rovnici pomocí Laplaceovy transformace:

$$y''(t) + 4y'(t) + 4y(t) = u(t) \quad \omega > 0, y(0) = 0, y'(0) = 0, u(t) = 3 \quad (84)$$

Rovnice je opět převedena na obrazy přes Laplaceovu transformaci:

$$\begin{aligned}
 y''(t) + 4y'(t) + 4y(t) &= u(t) & (85) \\
 s^2 Y(s) + 4sY(s) + 4Y(s) &= \frac{3}{s} \\
 Y(s)(s^2 + 4s + 4) &= \frac{3}{s} \\
 Y(s) &= \frac{3}{s(s^2 + 4s + 4)} = \frac{3}{s(s+2)^2} = \frac{3}{4s} - \frac{3}{4(s+2)} - \frac{3}{2(s+2)^2} \\
 y(t) &= \underline{\underline{\frac{3}{4} - \frac{3}{4}e^{-2t} - \frac{3}{2}te^{-2t}}}
 \end{aligned}$$

## 9 METODA POŽADOVANÉHO MODELU

Prezentace 5. Popis uzavřeného regulačního obvodu.ppt byla doplněna o témata kvalita regulace a metoda požadovaného modelu. Téma kvalita regulace obsahuje kapitoly, které stručně vysvětlují ukazatele kvality a integrální kritéria. Hlavní důraz je kladen na popis metody požadovaného modelu. Prezentace obsahuje jak kompletní tabulku Tabulka 7: Hodnoty stavitelných parametrů konvenčních regulátorů pro metodu MPM, tak i tabulku Tabulka 6: Závislost koeficientů  $\alpha$  a  $\beta$  na relativním překmitu  $\kappa$ . Na konci kapitoly je uveden vzorový příklad.

### 9.1 Metoda požadovaného modelu – řešený příklad

Je zadána regulovaná soustava s dopravním zpožděním:

$$G_s(s) = \frac{5}{(3s+1)(2s+1)} e^{-10s} \quad (86)$$

Úkolem je pro tuto soustavu navrhnout analogový ( $T=0$ ) regulátor tak, aby byl relativní překmit  $\kappa = 0,2$  (20%).

Pokud jsou uvažovány přibližné vztahy, tak tvar přenosu regulované soustavy je totožný s tvarem na řádce 8 v tabulce Tabulka 7: Hodnoty stavitelných parametrů konvenčních regulátorů pro metodu MPM. Pak z přenosu zadané regulované soustavy vyplývají následující vztahy:

$$\begin{aligned} k_1 &= 5 \\ T_1 &= 3 \\ T_2 &= 2 \\ T_d &= 10 \end{aligned} \quad (87)$$

Podle vztahů uvedených v tabulce Tabulka 7: Hodnoty stavitelných parametrů konvenčních regulátorů pro metodu MPM se vypočítají stavitelné parametry  $T_I^*$  a  $T_D^*$ .

$$\begin{aligned} T_I^* &= T_1 + T_2 = 3 + 2 = 5 \\ T_D^* &= \frac{T_1 T_2}{(T_1 + T_2)} = \frac{3 \cdot 2}{(3 + 2)} = 1,2 \end{aligned} \quad (88)$$

V závislosti na požadovaném relativním překmitu  $\kappa$  se v tabulce Tabulka 6: Závislost koeficientů  $\alpha$  a  $\beta$  na relativním překmitu  $\kappa$  určí koeficient  $\beta$  a vypočítá se koeficient  $a$ .

Následně se opět podle vztahu z tabulky Tabulka 7: Hodnoty stavitelných parametrů konvenčních regulátorů pro metodu MPM určí doporučené zesílení regulátoru  $k_R^*$ .

$$a = \frac{1}{\beta T_d} = \frac{1}{1,437 \cdot 10} \doteq 0,0696 \quad (89)$$
$$k_R^* = \frac{a T_I^*}{k_1} = \frac{0,0696 \cdot 5}{5} = 0,0696$$

Doporučený regulátor bude mít tvar přenosu:

$$G_R(s) = 0,0696 \left( 1 + \frac{1}{5s} + 1,2s \right) \quad (90)$$

## 10 APLIKACE NA VYKRESLOVÁNÍ GRAFŮ

V prezentaci 5. Popis uzavřeného regulačního obvodu.ppt byly přidány dva nové snímky, které obsahují vložené objekty Proporcionalni\_clen.jar a Integracni\_clen.jar. Tyto dva objekty se spustí kliknutím myši. Po jejich spuštění se objeví dialogové okno, které vyzívá k vložení koeficientů pro daný přenos. Po vložení platných hodnot a kliknutí na tlačítko vykreslit, aplikace začne animačně vykreslovat přechodové charakteristiky zadaných přenosů.

### 10.1 Proporcionální člen

Byl vytvořen projekt Proporcionalni\_2\_rad\_komplex, ve kterém byl vytvořen hlavní balíček main. Dále byly připojeny k tomuto projektu externí knihovny jcommon-1.0.16.jar a jfreechart-1.0.11.jar. Tyto knihovny nejsou standardně obsažené ve vývojovém prostředí Eclipse 3.5, a proto musely být importovány.

Hlavní balíček main obsahuje základní třídu LineChartDemo2.java, která se spouští jako první. Dále balíček main obsahuje třídy MojeOkno0rad.java, MojeOkno1.java, MojeOkno1rad.java, MojeOknoVse.java, Napoveda.java, Oaplikaci.java, animace0rad.java, animace1.java, animace1rad.java, animaceVse.java, kontrola.java. Vzhledem k velké rozsáhlosti zdrojových kódů všech tříd budou popsány pouze nejdůležitější bloky.

LineChartDemo2.java je třída, která slouží k vykreslování okna s průběhem grafu. Tato třída zajišťuje správné nastavení os, popisků, legendy.

Zdrojový kód této třídy:

```
1      package main;
2      import java.awt.BorderLayout;
3      import java.awt.Color;
4      import org.jfree.chart.ChartFactory;
5      import org.jfree.chart.ChartPanel;
6      import org.jfree.chart.JFreeChart;
7      import org.jfree.chart.axis.NumberAxis;
8      import org.jfree.chart.plot.PlotOrientation;
9      import org.jfree.chart.plot.XYPlot;
10     import org.jfree.chart.renderer.xy.XYLineAndShapeRenderer;
11     import org.jfree.data.xy.XYDataset;
```

```
12     import org.jfree.data.xy.XYSeries;
13     import org.jfree.data.xy.XYSeriesCollection;
14     import org.jfree.ui.ApplicationFrame;
15     import org.jfree.ui.RectangleInsets;
```

Na začátku zdrojového kódu jsou importovány všechny potřebné knihovny, které tato třída bude používat. Zdrojový kód pokračuje:

```
16     public class LineChartDemo2 extends ApplicationFrame {
17         private static final long serialVersionUID = 1L;
18         static MojeOkno1 okno1 = new MojeOkno1();
19         static MojeOkno1rad okno3 = new MojeOkno1rad();
20         static MojeOkno okno2 = new MojeOkno();
21         static MojeOkno0rad okno0 = new MojeOkno0rad();
22         static MojeOknoVse oknoVse = new MojeOknoVse();
23         static Napoveda oknoNapoveda = new Napoveda();
24         static Oaplikaci oknoOApplikaci = new Oaplikaci();
25         XYDataset dataset ;
26         public XYSeries series1; //G1
27         public XYSeries series2; //G2
28         public XYSeries series3; //G3
29         public XYSeries series4; //K1
30         public XYSeries series5; //K2
31         public XYSeries series6; //K3
32         static String popisGrafu;
```

Byly deklarovány všechny proměnné, které tato třída bude využívat. Zdrojový kód pokračuje vytvořením konstruktoru:

```
33     public LineChartDemo2(String title) {
34         super(title);
35         popisGrafu = title;
36         dataset = createDataset();
37         JfreeChart chart = createChart(dataset);
38         ChartPanel chartPanel = new ChartPanel(chart);
39         chartPanel.setPreferredSize(new java.awt.Dimension(500,
40             270));
41         BorderLayout l=new BorderLayout();
42         chartPanel.setLayout(l);
43         setContentPane(chartPanel);
44     }
```

V konstruktoru jsou vytvořeny základní objekty, které reprezentují jednotlivé části grafu. Velmi důležité je volání metody `createDataset()`, která reprezentuje data grafu. Proměnná

popisekGrafu obsahuje řetězec, který byl předán konstuktoru při jeho volání. Zdrojový kód pokračuje definicí metody createDataset():

```
44     private XYDataset createDataset() {
45         if(popisekGrafu.equals(„Proporcionální“) series1 = new
XYSeries(„1.řád“);
46         else series1 = new XYSeries(„G1“);
47
48         if(popisekGrafu.equals(„Proporcionální“) series2 = new
XYSeries(„2.řád real“);
49         else series2 = new XYSeries(„G2“);
50
51         if(popisekGrafu.equals(„Proporcionální“) series3 = new
XYSeries(„2.řád komplex“);
52         else series3 = new XYSeries(„G3“);
53
54         if(popisekGrafu.equals(„Proporcionální 1.řádu“)
series4 = new XYSeries(„K1“);
55         else {
56             if(popisekGrafu.equals(„Proporcionální“) series4 = new
XYSeries(„0.řád“);
57             else series4 = new XYSeries(„K“);}
58
59         series5 = new XYSeries(„K2“);
60         series6 = new XYSeries(„K3“);
61
62         XYSeriesCollection dataset = new XYSeriesCollection();
63         dataset.addSeries(series1);
64         dataset.addSeries(series2);
65         dataset.addSeries(series3);
66         if(popisekGrafu.equals(„Proporcionální 1.řádu“)){
67             dataset.addSeries(series4);
68             dataset.addSeries(series5);
69             dataset.addSeries(series6);
70         }
71         if(popisekGrafu.equals(„Proporcionální 2.řádu,
komplexní kořeny“) || popisekGrafu.equals(„Proporcionální 2. řádu,
reálné kořeny“) || popisekGrafu.equals(„Proporcionální“))
dataset.addSeries(series4);
72
73         return dataset;
74     }
```

Metoda `createDataset()` vytvoří na základě proměnné `popisekGrafu` jednotlivé řady grafu s příslušnými popiskami. Vytvořené řady se sloučí do jediného objektu kolekce řad s názvem `dataset` a tato kolekce je vrácena příkazem `return`. Zdrojový kód pokračuje metodou `createChart()` pro vytvoření vlastního grafu:

```
75     private static JfreeChart createChart(XYDataset dataset) {
76         JfreeChart chart = ChartFactory.createXYLineChart(
77             popisekGrafu, // chart title
78             „t“, // x axis label
79             „y“, // y axis label
80             dataset, // data
81             PlotOrientation.VERTICAL,
82             true, // include legend
83             true, // tooltips
84             false // urls
85         );
86
87         // NOW DO SOME OPTIONAL CUSTOMISATION OF THE CHART...
88         chart.setBackgroundPaint(Color.white);
89         XYPlot plot = (XYPlot) chart.getPlot();
90
91         plot.setBackgroundPaint(Color.white);
92         // get a reference to the plot for further customisation...
93         plot.setAxisOffset(new RectangleInsets(5.0, 5.0, 5.0,
94             5.0));
95         plot.setDomainGridlinePaint(Color.white);
96         plot.setRangeGridlinePaint(Color.white);
97         XYLineAndShapeRenderer renderer
98             = (XYLineAndShapeRenderer) plot.getRenderer();
99         renderer.setShapesVisible(false);
100        renderer.setShapesFilled(false);
101
102        //renderer.setSeriesPaint(0, Color.black);
103        //renderer.setSeriesPaint(1, Color.black);
104        renderer.setSeriesPaint(2, new Color(255, 102, 0));
105        renderer.setSeriesPaint(3, Color.black);
106        renderer.setSeriesPaint(4, new Color(0, 204, 0));
107        renderer.setSeriesPaint(5, new Color(255, 204, 0));
108
109        // utor the auto tick unit selection to integer units
110        only...
```

```

109         NumberAxis     rangeAxis     =     (NumberAxis)
plot.getRangeAxis();
110
    rangeAxis.setStandardTickUnits(NumberAxis.createIntegerTickUnits())
    ;
111         // OPTIONAL CUSTOMISATION COMPLETED.
112         return chart;
113     }

```

Metoda `createChart()` má jako parametr již zmíněnou kolekci řad dataset. V této metodě je vytvořen vlastní objekt grafu. Obsahuje také metody, které nastavují vzhled grafu. Metoda vrátí objekt grafu `chart`. [11], [12]

Zdrojový kód pokračuje hlavní metodou `main()`:

```

114     public     static     void     main(String[]     args)     throws
InterruptedException {
115         okno0.show();
116     }

```

Hlavní metoda `main()`, pouze zobrazí `okno0`, které je typu `MojeOkno0rad`. Objekt `okno0` reprezentuje vizuální okno, ve kterém je uživatel vyzván k vložení koeficientů. Aplikace je vytvořená tak, aby se metoda `main()` v této třídě volala pouze jednou při prvním spuštění aplikace. Zdrojový kód pokračuje metodou `zobrazeni()`:

```

117         public void zobrazeni(int okno){
118             if(okno == 2){           //komplexni koreny - 2.řád
119                 okno1.show();
120                 okno2.hide();
121                 okno3.hide();
122                 okno0.hide();
123                 oknoNapoveda.hide();
124                 oknoOAplikaci.hide();
125                 oknoVse.hide();
126             }
127             if(okno == 22){ //realne koreny - 2. řád
128                 okno2.show();
129                 okno1.hide();
130                 okno3.hide();
131                 okno0.hide();
132                 oknoNapoveda.hide();
133                 oknoOAplikaci.hide();
134                 oknoVse.hide();
135             }

```

```
136         if(okno == 1){           // 1.řád
137             okno3.show();
138             okno1.hide();
139             okno2.hide();
140             okno0.hide();
141             oknoNapoveda.hide();
142             oknoOAplikaci.hide();
143             oknoVse.hide();
144         }
145
146         if(okno == 0){           // 0.řád
147             okno0.show();
148             okno1.hide();
149             okno2.hide();
150             okno3.hide();
151             oknoNapoveda.hide();
152             oknoOAplikaci.hide();
153             oknoVse.hide();
154         }
155
156         if(okno == 3){           // Napoveda
157             oknoNapoveda.show();
158             okno1.hide();
159             okno2.hide();
160             okno3.hide();
161             okno0.hide();
162             oknoOAplikaci.hide();
163             oknoVse.hide();
164         }
165
166         if(okno == 4){           // 0 aplikaci
167             oknoNapoveda.hide();
168             okno1.hide();
169             okno2.hide();
170             okno3.hide();
171             okno0.hide();
172             oknoOAplikaci.show();
173             oknoVse.hide();
174         }
175
176         if(okno == 5){           // Vse
```

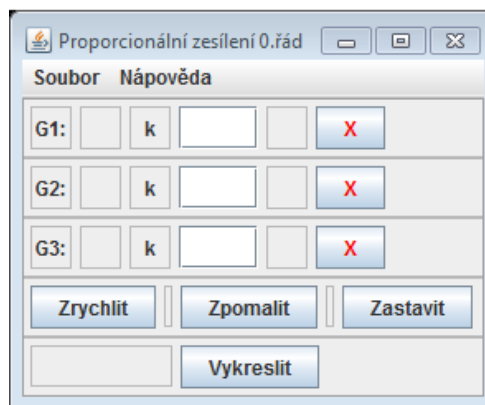
```

177         oknoNapoveda.hide();
178         okno1.hide();
179         okno2.hide();
180         okno3.hide();
181         okno0.hide();
182         oknoOAplikaci.hide();
183         oknoVse.show();
184     }
185 }
186 }

```

Metoda zobrazeni() zobrazí podle parametru, který je jí předán, příslušný objekt a ostatní objekty skryje. Jednotlivé objekty reprezentují vizuální okna, které slouží k zadávání koeficientů jednotlivých přenosů daných řádů. [11], [12]

Po prvním spuštění aplikace je zobrazeno okno pro 0. Řád přenosu. Okno vypadá následovně:



Obrázek 21: Proporcionální zesílení  
0.řád

Vytvoření grafických prvků proběhlo pomocí funkce drag&drop, kterou vývojové prostředí Eclipse 3.5 umožňuje. Přetažením požadovaných prvků na formulář je vygenerován kód příslušných grafických objektů. Pravým kliknutím na grafický objekt je vybrána příslušná metoda obsluhující prvek. Kvůli značné rozsáhlosti kódu budou vysvětleny jen přidávané metody:

Zdrojový kód metody tlačítka s popiskem Zrychlit:

```

1         jButton1.addActionListener(new
        java.awt.event.ActionListener() {
2             public void actionPerformed(java.awt.event.ActionEvent
        e) {

```

```

3         if(rychlost-30>2) {
4             rychlost= rychlost - 30;
5             jButton1.setText(„Zrychlit“);
6             jButton1.setBackground(jButton2.getBackground());
7         } else {
8             jButton1.setText(„ Max!!!“);
9             jButton1.setBackground(new Color(255, 0, 0));
10            jButton1.setPreferredSize(new Dimension(76,26));
11        }
12    }
13    });

```

Tato metoda zajistí, že při kliknutí na tlačítko jButton1 (Zrychlit) dojde ke snížení proměnné rychlost. Tato proměnná se sníží pouze pokud je splněna podmínka. Pokud podmínka splněna není, nastaví se pozadí tlačítka na červenou barvu a změní se popisek tlačítka.

Metoda tlačítka jButton2 s popisem Zpomalit je velmi podobná předcházející metodě. Rozdíl je pouze ve zvyšování proměnné rychlost. Pokud je proměnná rychlost menší než hodnota 10, tak dojde ke změně popisku tlačítka jButton1 (Zrychlit).

Proměnná rychlost bude dále v programu využívána ke zrychlování nebo zpomalování animace grafu.

Tlačítko jButton3 (Zastavit) má za úkol nastavit hodnotu proměnné zastavit na true. Pomocí této proměnné se zastavuje aktuální vykreslování grafu.

```

1     public void actionPerformed(java.awt.event.ActionEvent
2     e) {
3         zastavit = true;
4     }

```

Tlačítka s popisem X slouží k vymazání hodnoty v textových polích a vymazání dat jednotlivých řad grafu. Jednoduchý zdrojový kód jeho metody:

```

1     public void actionPerformed(java.awt.event.ActionEvent
2     e) {
3         noveokno0rad.series1.clear();
4         jTextFieldK1.setText(„“);

```

```
4           }
```

Nejdůležitější je tlačítko s popiskem Vykreslit. Po kliknutí na toto tlačítko dojde k několika operacím a následnému zobrazení okna s animací grafu. Vzhledem k opakování principu jednotlivých bloků zdrojového kódu budou uvedeny pouze některé části kódu:

```
1     public void actionPerformed(java.awt.event.ActionEvent e) {
2
3         if (animace0rad.bezi){
4             zastavit = true;
5         }else{
6             String text;
7     try{
8         prevodOKk1=kontrola.zkontrolovat(jTextFieldK1);
9         text = jTextFieldK1.getText().replaceAll(",",".");
10        if(prevodOKk1) K1=Float.valueOf(text);
11            else K1 =0;
12        }
13    catch(Exception eK){
14        System.out.println(„Musíš zadat platné desetinné číslo K1 „);
15        System.out.println(„Vygenerována výjimka „+eK);
16        prevodOKk1=false;
17    }
18    noveokno0rad.setSize(500,300);
19    noveokno0rad.setVisible(true);
20    RefineryUtilities.centerFrameOnScreen(noveokno0rad);
21    counter = -1;
22    if(pomocna!=K1 && prevodOKk1){ //byla provedena zmena
    v textovem poli kl
23        noveokno0rad.series1.clear();
24        vykresleni=true; //byla zmena, musime znova vykreslit
25        pomocna=K1; //zda byla zmenena promenna
26    }
27    if(vykresleni || vykresleni2 || vykresleni3){
28        animace0rad an=new animace0rad();
29        an.mmain(null);
30        an = null;
31    }
32    }}
```

Pokud proměnná `bezi` je rovna hodnotě `true`, tak se zastaví aktuální vykreslování grafu. V opačném případě, kdy se žádný graf nevykresluje, dojde ke kontrole vstupních údajů

v textových polích pomocí třídy kontrola. Dále je nastavena velikost, viditelnost, zarovnání nového okna, které vykreslí graf. Pomocná proměnná pomocna slouží k uchování informace, zda byla změněna hodnota v textovém poli. Pokud změněna nebyla, tak se nebude graf pro toto textové pole znovu vykreslovat. Samotné vykreslení grafu probíhá v objektu typu animace0rad. Tato třída bude popsána po třídě kontrola:

```

1      public static boolean zkontrolovat(JTextField textBox) {
2          String text;
3          text = textBox.getText();
4          text = text.replaceAll(",",".");
5
6          String expression = „^[+-]?[0-9]*\\.?[0-9]*$“;
7          CharSequence inputStr = text;
8          Pattern pattern = Pattern.compile(expression);
9          Matcher matcher = pattern.matcher(inputStr);
10         if(matcher.matches() && Integer.valueOf(text.length()) > 0){
11             textBox.setBackground(Color.white);
12             return true;
13         }else { if(Integer.valueOf(text.length()) > 0)
14             textBox.setBackground(new Color(255, 32, 0));
15             else textBox.setBackground(Color.white);
16             return false;
17         }
18     }

```

Třída kontrola obsahuje metodu zkontrolovat, ve které se pomocí regulárních výrazů vyhodnotí správnost zadaného řetězce. V případě nevhodných zadaných znaků (písmena, dvojtečky,...) dojde ke zbarvení textboxu. Metoda vrací hodnotu true (správně zadaný řetězec) nebo false (chybně zadaný řetězec).

Třída animace0rad definuje metody na vykreslení vlastního grafu. Budou popsány jen nejdůležitější bloky, jejichž princip se opakuje pro každý graf.

```

1 package main;
2 public class animace0rad implements Runnable {
3     public static Thread t2;
4     static boolean probehlo0rad=false;
5     double krok1=0,krok2=0,krok3=0;
6     static boolean bezi=false;
7
8     public animace0rad() {

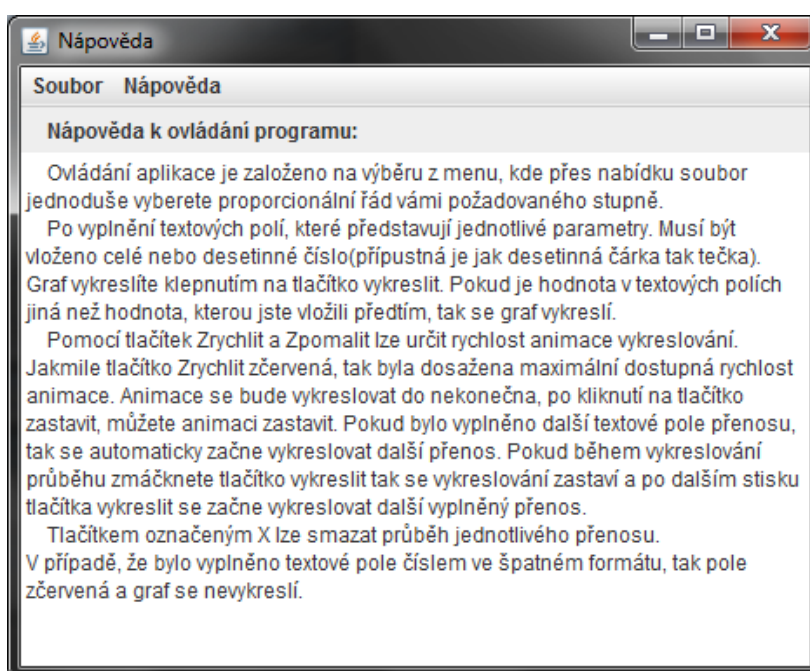
```

```
9         t2 = new Thread(this);
10     }
11
12     public void run(){
13         if(MojeOkno0rad.vykresleni){
14             krok1 = MojeOkno0rad.K1/10;
15             bezi = true;
16             while(MojeOkno0rad.zastavit == false){
17                 try {
18                     Thread.sleep((long)MojeOkno0rad.rychlost);
19                 } catch (InterruptedException e1) {
20                     e1.printStackTrace();
21                 }
22                 if(MojeOkno0rad.counter<0){MojeOkno0rad.noveokno0rad.series1.
                add(MojeOkno0rad.counter, 0);}
23                 if(MojeOkno0rad.counter + krok1 > 0 && MojeOkno0rad.counter <
                0 || MojeOkno0rad.counter == 0){
24                     MojeOkno0rad.noveokno0rad.series1.add(0,0);
25                     MojeOkno0rad.noveokno0rad.series1.add(0,MojeOkno0rad.K1);
26                 }
27                 if(MojeOkno0rad.counter>0){MojeOkno0rad.noveokno0rad.series1.
                add(MojeOkno0rad.counter, MojeOkno0rad.K1);}
28                 MojeOkno0rad.counter=MojeOkno0rad.counter+krok1;
29             }
30
31             MojeOkno0rad.zastavit = false;
32             MojeOkno0rad.vykresleni=false;
33             MojeOkno0rad.counter=-1;
34             bezi = false;
35             }//while counter<5
36             probehlo0rad = true;
37         }
38     public void mmain(String[] args) {
39         t2.start();
40         t2=null;
41     }
42     }}
```

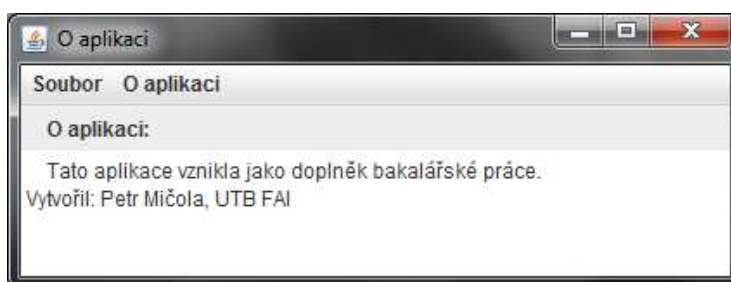
V konstruktoru třídy animace0rad je vytvořeno nové vlákno. Metoda mmain() vlákno spustí. V těle vlákna jsou definovány pomocné proměnné. Pokud došlo ke změně hodnoty v některém z textových polí, tak je vykreslen příslušný graf. U proporcionálního zesílení 0. Řádu je krok pro výpočet desetinnou zesílení. Během vykreslování grafu je nastaveno proměnná bezi na hodnotu true. Před vykreslením dalšího bodu do grafu je vždy vlákno

uspáno na dobu, která odpovídá proměnné rychlost. Následuje výpočet dat pro jednotlivé řady grafu a tyto data jsou přidány do grafu. Vykreslování grafu probíhá dokud je hodnota proměnné zastavit rovna false. Po zastavení animace jsou resetovány pomocné proměnné a proměnná proběhlo0rad nastavena na true. Tato proměnná slouží k indikaci skutečnosti, že už došlo alespoň k jednomu vykreslení grafu (následně se již nemusí vytvářet nové okno pro graf, ale nová data se zapisují do již vytvořeného objektu grafu).

Třídy Oaplikaci a Napoveda obsahují pouze textová pole, s informacemi o aplikaci. Je zde popsáno ovládání a funkce programu.

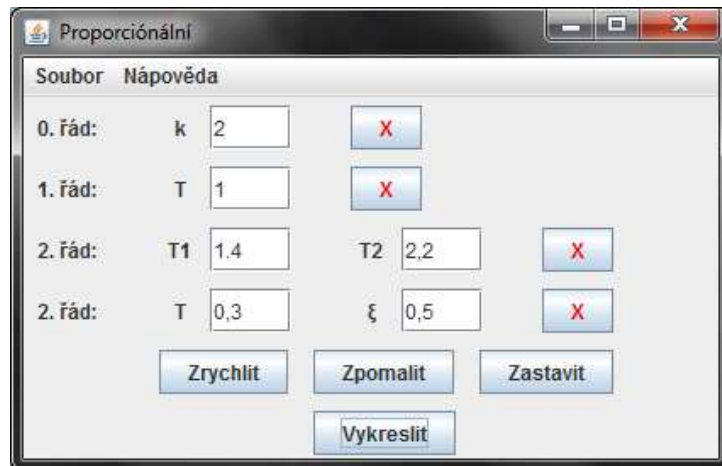


Obrázek 22: Dialogové okno nápověda

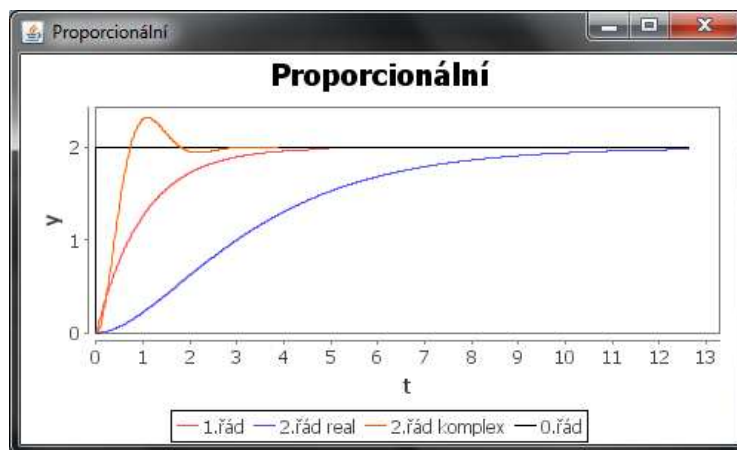


Obrázek 23: Dialogové okno o aplikaci

Zdrojové kódy zbývajících tříd mají velmi podobnou strukturu jako právě popsané třídy. Odlišností je pouze počet vstupních textových polí a výpočet funkční hodnoty. Třídy MojeOknoVse a animaceVse vykreslují přenosy 0. Řádu, 1. Řádu, 2. Řádu v jednom grafu.



Obrázek 24: Dialogové okno proporcionální



Obrázek 25: Dialogové okno proporcionální graf

## 10.2 Integrovní člen

Aplikace pro vykreslování přechodových charakteristik integračních členů je principiálně totožná s aplikací na vykreslování přechodových charakteristik proporcionálních členů. Rozdílem je pouze vzorec pro výpočet funkční hodnoty.

## ZÁVĚR

Hlavním cílem této bakalářské práce bylo rozšířit a doplnit již existující prezentace, které slouží jako pomůcka pro přednášky předmětu Automatizace. Byly doplněny nové kapitoly, jejichž teoretický základ je popsán v teoretické části této práce. Do zvolených kapitol byly vloženy vzorové řešené příklady, které mají za úkol usnadnit pochopení přednášených témat.

Nejrozsáhlejším blokem v této práci je popis zdrojového kódu vytvořené aplikace, které animuje přechodové charakteristiky zadaných systémů. Vzhledem k velmi rozsáhlému projektu byly popsány pouze nejzákladnější třídy a jejich metody. Ostatní třídy jsou velmi podobné a pracují na obdobném principu. K vytvoření této aplikace byl využit programovací jazyk Java, který nabízí široké možnosti. Hlavní jeho výhodou je jednoduchost, přenositelnost a velká komunita lidí, kteří spolu v internetových diskuzích komunikují a řeší vzniklé problémy. Z tohoto důvodu jsem si zvolil tento programovací jazyk k vytvoření požadované aplikace. Důležitým krokem bylo zvolení vhodné knihovny, která by usnadnila vytvoření grafů a zároveň by poskytovala široké možnosti při jejich návrhu. Po krátkém internetovém průzkumu jsem si zvolil knihovnu JfreeChart, která je zcela zdarma. Tato knihovna nabízí opravdu velké množství funkcí, kterými lze vytvořit přehledné a moderní grafy. Hlavní nevýhodou této knihovny je fakt, že vzorové příklady jsou placené. Ale i přesto bych tuto knihovnu doporučil jako vhodný nástroj při vytváření grafů v Javě. Velkou výhodou této knihovny je možnost použití vláken. V mém projektu jsem vlákna využil k animaci přechodových charakteristik. Animace zvyšuje pozornost a zájem u studentů, což byl jeden z důvodů zavedení animací do aplikace. Díky této aplikaci si můžou studenti vykreslit a porovnat přechodové charakteristiky pro různé koeficienty, které si sami zvolí.

Během testování aplikace jsem nenarazil na žádný závažnější problém. Nicméně bych tuto aplikaci doporučoval používat pouze jako ilustrativní pomůcku při studiu.

## ZÁVĚR V ANGLIČTINĚ

The main goal of the bachelor thesis was to extend existing presentations which are used like in the subject of Automation. New topics were added. Theoretical basic is described in theoretical part of this thesis. Illustrative solved examples were inserted into selected chapters. The task of the solved examples is to help and simplify lectured topics.

The largest block in the thesis is description of source codes. Created application animates step response of inserted systems. Only basic classes and their methods were described due to very large source codes of the project. Other classes are very similar and work on analogical principle. The application was created in programming language Java which offers many features and functions. Main advantages of Java are simplicity, portability and big community of people who communicate in internet discussions and deal with their problems. It was the main reason why I chose this programming language to create required application. Choice of the library was very important. I needed library which would help me to create various types of graphs with many preferences. I chose JfreeChart library after short internet research. It is free and it offers my requirements. I can create clear and modern graphs with this library. Main disadvantage of the library are paid examples. But in spite of that I would recommend it. It is appropriate tool to create graphs in Java. Threads can be used, which is big advantage. I used it in my project to animate step response. Animation increases attention and interest of students. It was one of the reasons why I inserted animation into application. Students can simulate and compare step response for various coefficients due to application.

I haven't found any serious problems during testing application. In spite of that I recommend to use it only like a help tool.

## SEZNAM POUŽITÉ LITERATURY

Monografie:

- [1] BALÁTEĚ, J. *Automatické řízení*. 1. Vydání. Praha : BEN – technická literatura, 2003. 664 s. ISBN 80-7300-020-2.
- [2] FRANKLIN, G. F.; POWELL, J. D.; EMAMI-NAEINI, A. *Feedback control of dynamic systems*. 5th edition. New Jersey : Pearson Education, Inc., 2006. 910 s. ISBN 0-13-149930-0.
- [3] KEOGH, J. *Java bez předchozích znalostí : Průvodce pro samouky*. 1. Vydání. Brno : CP Books, a.s., 2005. 274 s. ISBN 80-251-0839-2.
- [4] PROKOP, R.; MATUŠŮ, R.; PROKOPOVÁ, Z. *Teorie automatického řízení : lineární spojité dynamické systémy*. 1. Vydání. Zlín : UTB FAI ve Zlíně, 2006. 102 s. ISBN 80-7318-369-2.
- [5] SPELL, B. *Pro Java Programming*. 2nd edition. New York : Apress, 2005. 703 s. ISBN 1-59059-474-6.
- [6] ŠVARC, I. *Automatizace : Automatické řízení*. Brno : Akademické nakladatelství CERM, s.r.o. Brno, 2002. 201 s. ISBN 80-214-2087-1.
- [7] ŠVEC, J. ; KOTEK, Z., et al. *Teorie automatického řízení*. Praha : SNTL – Státní nakladatelství technické literatury, 1969. 461 s. ISBN 04-007-69.
- [8] VAŠEK, V. *Teorie automatického řízení II*. 1. Vydání. Brno : Rektorát VUT v Brně, 1990. 139 s. ISBN 80-214-0115-X.
- [9] VÍTEČKOVÁ, M. *Seřízení regulátorů metodou inverze dynamiky*. 1. Vydání. Ostrava : VŠB – Technická univerzita Ostrava, 1998. 56 s.
- [10] VÍTEČKOVÁ, M.; VÍTEČEK, A. *Základy automatické regulace*. Přepřacované 2.vydání. Ostrava : VŠB – Technická univerzita Ostrava, 2008. 244 s. ISBN 978-80-248-1924-2.

## Internetové zdroje:

- [11] *Programming tutorials and source code examples* [online]. 2009 [cit. 2011-05-28]. JfreeChart: XY Line And Shape Renderer Demo. Dostupné z WWW: <<http://www.java2s.com/Code/Java/Chart/JFreeChartXYLineAndShapeRendererDemo.htm>>.
- [12] *Screaming Penguin | - Source Does Matter* [online]. 2005 [cit. 2011-05-28]. JfreeChart utoriál (reprinted) | Screaming Penguin. Dostupné z WWW: <<http://www.screaming-penguin.com/node/4005>>.
- [13] SMUTNÝ, P. *E-Automatizace : Informační portál z oblasti automatizace* [online]. 2009 [cit. 2011-05-26]. Dostupné z WWW: <<http://www.e-automatizace.cz>>.

## Interní materiály:

- [14] ŠVEJDA, J. *Multimediální podpora výuky předmětu Automatizace*. Zlín : UTB FAI ve Zlíně, 2009.
- [15] VAŠEK, V. *Podklady pro přednášky z předmětu Mikropočítače : Interní pomůcka*. Zlín : FT UTB ve Zlíně, 2005. 222 s.

**SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK**

URO        Uzavřený regulační obvod.

MPM        Metoda požadovaného modelu.

JVM        Java Virtual Machine.

GUI        Graphical user interface.

WWW       World Wide Web

PC         Personal Computer

**SEZNAM OBRÁZKŮ**

Obrázek 1: Karnaughova mapa pro tři vstupní proměnné .....	14
Obrázek 2: Základní pravidla Booleovy algebry .....	15
Obrázek 3: Karnaughova mapa funkce $y=x_2x_3$ .....	17
Obrázek 4: Geometrická interpretace linearizace tečnou .....	23
Obrázek 5: Proporcionální členy – přechodové charakteristiky .....	25
Obrázek 6: Integrovaní členy – přechodové charakteristiky .....	27
Obrázek 7: Derivační členy – přechodové charakteristiky .....	28
Obrázek 8: Odezvy regulačního obvodu na skokové změny: a) žádané veličiny $w(t)$ , b) poruchové veličiny $v(t)$ působící na výstupu regulované soustavy v případě nulových trvalých regulačních odchylek .....	31
Obrázek 9: Odezvy regulačního obvodu na skokové změny: a) žádané veličiny $w(t)$ , b) poruchové veličiny $v(t)$ působící na vstupu proporcionální regulované soustavy v případě regulátoru s integrační složkou .....	32
Obrázek 10: Přechodové charakteristiky kmitavého a nekmitavého přenosu s ukazateli kvality .....	33
Obrázek 11: Příklad regulační plochy .....	34
Obrázek 12: Lineární regulační plocha .....	35
Obrázek 13: Absolutní regulační plocha .....	36
Obrázek 14: Kvadratická regulační plocha .....	36
Obrázek 15: Přechodová charakteristika nekmitavého regulačního obvodu .....	39
Obrázek 16: Přechodové charakteristiky regulačního obvodu .....	39
Obrázek 17: Ukázka vývojového prostředí Eclipse 3.5 .....	49
Obrázek 18: Karnaughova mapa .....	52
Obrázek 19: Tabulka seskupených dvojic .....	53
Obrázek 20: Mřížka prostých implikantů .....	53
Obrázek 21: Proporcionální zesílení 0.řád .....	70
Obrázek 22: Dialogové okno nápověda .....	75
Obrázek 23: Dialogové okno o aplikaci .....	75
Obrázek 24: Dialogové okno proporcionální .....	76
Obrázek 25: Dialogové okno proporcionální graf .....	76

**SEZNAM TABULEK**

Tabulka 1: Pravdivostní tabulky logické funkce jedné proměnné .....	11
Tabulka 2: Pravdivostní tabulky logických funkcí OR, AND, NOR, NAND .....	12
Tabulka 3: Blokové schéma logických funkcí OR, AND, NOR, NAND, NON .....	13
Tabulka 4: Základní pravidla Booleovy algebry .....	16
Tabulka 5: Přenosy spojitých regulátorů .....	29
Tabulka 6: Závislost koeficientů $\alpha$ a $\beta$ na relativním překmitu $\kappa$ .....	39
Tabulka 7: Hodnoty stavitelných parametrů konvenčních regulátorů pro metodu MPM .....	40
Tabulka 8: Základní datové typy v jazyku Java .....	43
Tabulka 9: Nejčastěji používané metody vláken .....	47
Tabulka 10: Logická funkce – pravdivostní tabulka .....	51

## SEZNAM PŘÍLOH

P I Slovník Laplaceovy transformace

## PŘÍLOHA P I: SLOVNÍK LAPLACEOVY TRANSFORMACE

Originál $f(t)$	Obraz $F(s)$
$\delta(t)$	1
$1(t)$	$\frac{1}{s}$
$t^n$	$\frac{n!}{s^{n+1}}$
$e^{-at}$	$\frac{1}{s+a}$
$te^{-at}$	$\frac{1}{(s+a)^2}$
$\frac{1}{(n-1)!} t^{n-1} e^{-at}$	$\frac{1}{(s+a)^n}$
$1 - e^{-at}$	$\frac{a}{s(s+a)}$
$\frac{1}{a} (at - 1 + e^{-at})$	$\frac{a}{s^2(s+a)}$
$e^{-at} - e^{-bt}$	$\frac{b-a}{(s+a)(s+b)}$
$\sin \omega t$	$\frac{\omega}{s^2 + \omega^2}$
$\cos \omega t$	$\frac{s}{s^2 + \omega^2}$
$e^{-at} \sin \omega t$	$\frac{\omega}{(s+a)^2 + \omega^2}$
$e^{-at} \cos \omega t$	$\frac{s+a}{(s+a)^2 + \omega^2}$

[1], [2], [10]